LOGen: Toward LiDAR Object Generation by Point Diffusion

Ellington Kirby[1]
ellington.kirby@valeo.com
Mickael Chen[1]
mickael.chen.levinson@gmail.com
Renaud Marlet[1,2]
renaud.marlet@valeo.com
Nermin Samet[1]
nermin.samet@valeo.com

[1] Valeo.ai, Paris, France
[2] LIGM, ENPC, IP Paris, UGE, CNRS, France

Abstract

The generation of LiDAR scans is a growing topic with diverse applications to autonomous driving. However, scan generation remains challenging, especially when compared to the rapid advancement of image and 3D object generation. We consider the task of LiDAR object generation, requiring models to produce 3D objects as viewed by a LiDAR scan. It focuses LiDAR scan generation on a key aspect of scenes, the objects, while also benefiting from advancements in 3D object generative methods. We introduce a novel diffusion-based model to produce LiDAR point clouds of dataset objects, including intensity, and with an extensive control of the generation via conditioning information. Our experiments on nuScenes and KITTI-360 show the quality of our generations measured with new 3D metrics developed to suit LiDAR objects. The code is available at https://github.com/valeoai/LOGen.

1 Introduction

As 2D generative methods have exploded in their capacity to generate diverse and high quality images [30], one area of focus for 3D object generation has been understudied: objects in the context of LiDAR sensors. LiDAR is an essential modality for autonomous agents in driving scenarios. It provides accurate 3D information about the environment, making it crucial for safe navigation. The utility and corresponding shortcomings of LiDAR datasets [20, 26] makes the generation of LiDAR objects interesting from a variety of perspectives: sensor level generation for simulation [24], rare class augmentation [49], out-of-distribution generation [21], corner case generation [17], object disentanglement [28], etc.

Objects perceived by LiDAR sensors are distinct from classically rendered 3D objects in the impact that the relative angle and distance of objects to the sensor has on the final appearance of an object [51]. Further, LiDAR objects contain an additional channel of information when compared to classic 3D objects: the intensity of the laser beam reflected back to the sensor. Models which generate LiDAR objects must be capable of using this relative positioning and creating the intensity channel in order to generate realistic objects [41].

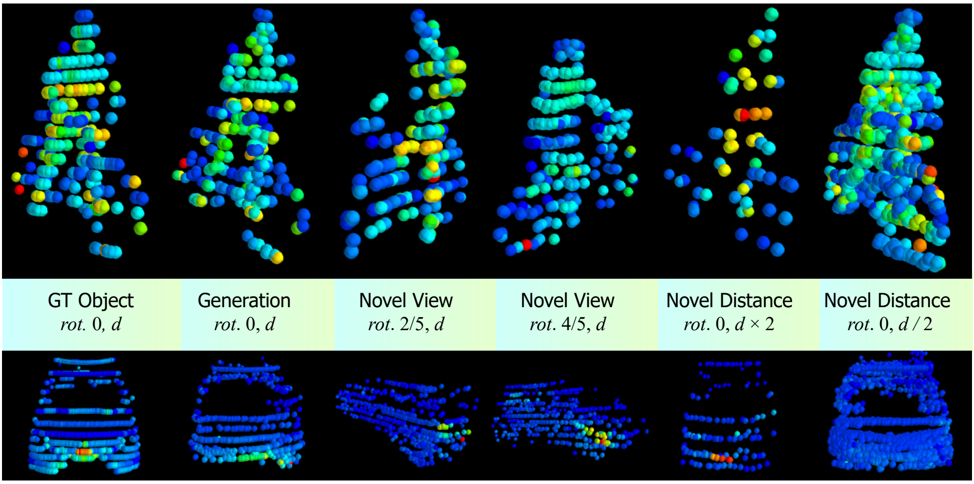

Figure 1: **Novel LiDAR objects generated by LOGen** trained on the nuScenes train set, produced using the conditioning information of real objects from the nuScenes validation set. Relative to the sensor, *rot* is the object's rotation while *d* is the distance. Point color is according to LiDAR intensity. More examples are in the supplementary material.

The generation of LiDAR data has been explored most extensively at the scene level, where many works have attempted to generate entire novel scans of autonomous vehicles in new surroundings [45, 51, 52]. However LiDAR scene generation remains a difficult task, with many works turning towards simplifying assumptions to focus the generative task onto more tractable footing [27, 29, 32, 42, 48]. We propose to bring recent advancements in generative methods [7, 25] to the domain of LiDAR data, focusing on the generation of high quality and realistic LiDAR objects.

We introduce a transformer-based diffusion model, dubbed LOGen (for LiDAR Object Generator), which is designed to generate LiDAR scans of new objects. The generation is parameterized by the object class, angle, and distance from which objects are seen by the LiDAR. This conditioning allows fine grained control of the appearance of generated objects.

We compare LOGen against suitable alternatives that we design, inspired from other tasks: a sparse 3D CNN (MinkUNet [8]); a 2D transformer-based generator (PixArt-α [7]) adapted to 3D; and a 3D transformer-based generator (DiT-3D [25]), originally designed for traditional 3D objects, which we adapt to LiDAR objects. We also compare to concurrent work [46]. We show the quality of our generated objects compared to these baselines. We also evaluate at scene level on the semantic segmentation task to assess the realism of objects we generate. Examples of generation are shown in Fig. 1. Our contributions are as follows:

- We propose a diffusion-based method specifically designed to produce LiDAR point clouds of objects, including intensity and with an extensive control of generation (conditioning).
- We propose a new transformer-based LiDAR object diffusion model, operating on 3D points, as well as three other diffusion-based baselines.
- Our experiments show that despite the compact size of our model (7.5M parameters), we

achieve superior quality, which we also measure with new metrics suited to LiDAR objects.

We believe that the LiDAR object generation task is ripe for future exploration, and that the promising results we present here will serve as a basis for this task.

2 Related Work

3D object generation. Various techniques have been proposed for 3D object generation: GANs [1, 35], VAEs [18, 40], normalizing flows [47], diffusion at point level [23, 38], and diffusion on features at voxel level with a transformer architecture [25]. However, these methods are only typically evaluated on datasets such as ShapeNet [6], ModelNet [43], or CO3D [33] after sampling points uniformly on shape surfaces, rather than following LiDAR sampling patterns. Their behavior with uneven point distributions of (sensor-specific) LiDAR scans is unexplored. Besides, these methods do not use geometric conditioning related to viewpoint and do not generate point intensities. Further, most of these methods are trained and evaluated with a high and fixed number of points: typically 2048. It is not clear how they would behave with varying and sparse point clouds, as is the case of LiDAR objects.

LiDAR data generation. Methods have been proposed to generate full LiDAR scans, by LiDAR simulation [9, 10, 11] in synthetic environments [48], or by learning to generate LiDAR data using energy-based models [51], VQ-VAEs [45], diffusion in range images [4, 15, 27, 32] or diffusion on 3D points [29], possibly with images [50]. Further, simulation and learning can be combined to improve realism [24, 52]. Some scan generation methods incorporate objects via: conditioning on a semantic map [32]; placing learned codes of object classes into a canvas representing the scene before decoding [45]; in-painting objects into range images [4]; and using text-based conditioning [42]. However these methods do not focus on the visual fidelity of generated objects, do not directly learn object appearance and intensity from the data at an object level, and do not evaluate the object perceptive quality.

OLiDM [46] is a concurrent work on LiDAR object generation that highlights the importance of distinguishing between foreground and background objects, employing a diffusion transformer model to generate LiDAR objects. They allow conditioning on object viewpoint parameters to produce localized objects, and perform scene level generation using the generated objects as conditions. LOGen in contrast focuses on architecture level decisions and uses novel methods of incorporating conditioning information and encoding points. LOGen features more robust evaluation of generated objects, with classwise metrics and perceptual evaluations including generated intensities.

3 Method

The goal of the *LiDAR object generation task* is to learn, for known classes in a labeled LiDAR dataset, to generate 3D point clouds of novel objects, as they would be captured by the same LiDAR sensor from arbitrary viewpoints. As opposed to simulation, the training distribution is given by real-world objects extracted from 3D scans.

We train a diffusion model to generate point clouds of novel objects. The diffusion process is applied directly on points given as input, and is conditioned by geometric information concerning the object to generate.

3.1 Object parameterization

An object in a LiDAR dataset is given by its class c, a set of 3D points with intensities, as well as information on the object bounding box, namely, the box center (x,y,z), the box length, width and height (l,w,h) and the box yaw ψ, i.e., the orientation around the vertical z axis (azimuth) of the heading of the object.

Parameters. To reduce the number and variability of input and conditioning parameters when training a model, we normalize the representation of objects with respect to the sensor location. For this, the Cartesian coordinates of object points are made relative to the oriented box center, i.e., relative to the box center and rotated around the z axis to align with the object box heading. The intensities are scaled according to the log-max.

We also decrease the number of parameters regarding the box center location and orientation as the object appearance, seen from the sensor, only depends on the distance d to the object and on the observation angle ϕ, i.e., the angle between the object heading ψ and the ray from the object bounding box center to the sensor. (As an illustration, $\phi=0$ means the object is facing the sensor, regardless of the object location around the ego vehicle.) In this conversion from Cartesian to cylindrical coordinates, the altitude z of the box center is kept.

In the end, the data distribution to learn consists of objects represented by a set of points $s=\{P_i\}_{i=1}^{N}$ in $\mathbb{R}^{N\times 4}$ with (centered) Cartesian coordinates and intensities. We do not normalize the point coordinates to preserve the absolute dimensions of objects and the specificity of the LiDAR scanning patterns, depending in particular on the distance to the sensor.

Last, we condition the generation on the following box information: $\kappa=(\phi,d,z,l,w,h)$. While our generation is classwise, class c is not directly included here as we train one model per class. This choice was made for compute and efficiency reasons: the per sample compute of single vs multi-class models is equivalent, thus allowing us to parallelize training. We nevertheless also experimented with a multi-class model, which was lagging behind the single-model-per-class approach in realism metrics (see supplementary material).

Point embedding. The way to embed points for diffusion has strong implications on the memory consumption and realism of generated outputs.

DiT-3D [25] introduces voxel embeddings that efficiently structure and aggregate points into patches. This technique, alongside window attention, allows the model to compute attention across shapes of thousands of points. The de-voxelization procedure places points in a final position inside their voxel, based on a weighted average of their activations. It has the effect of creating uniform and smooth point clouds across the underlying surface. Such uniform distributions are ideal for the goal of producing a continuous 3D mesh, but do not match the banded pattern observed when a LiDAR captures objects in the real world.

To address this issue, we introduce a simple embeddings based on PointNet [31]. These features are just the output of a single 1D convolution, with which we project from our input dimension (3+1) into the embedding dimension. Because the point clouds are in general magnitudes smaller (e.g., averaging from 40 points for bikes to 463 for bus in nuScenes) than the point clouds generated in the original DiT-3D paper (thousands of points), we can use standard memory-efficient attention layers and avoid the need for voxelization. In the experiment section, we examine the impact of both types of embeddings.

Conditioning encoding. A key stage when using diffusion is the encoding of the conditioning and time step. Classically, we rely on Fourier features [36], which involves a higher dimensional projection followed by a sine/cosine encoding of the corresponding features.

This applies to all scalar values of the condition but angle ϕ. To account for the periodic discontinuity of representation for an angle, we encode ϕ using the cyclical embeddings from [16], which preserves the cyclical nature of angular data. In our experiments, we train one network per class; we thus do not need to encode the class.

LiDAR object diffusion. Given a dataset with a number of object point clouds $S = \{s_j\}_{j=1}^{M}$ alongside conditions $K = \{\kappa_j\}_{j=1}^{M}$, the objective is to learn the conditional distribution $p(S|K)$. This is done via a Denoising Diffusion process: we learn to denoise input 4D Gaussian noise (N noisy points) into a target shape (N points representing a scanned object, with intensities).

We rely on the time-discrete formulation of the diffusion process known as Denoising Diffusion Probabilistic Models (DDPM) [14], which is also used in DIT-3D [25] and in LiDAR scene generation methods [29]. During training, we sample a random time step t and produce, via a forward noising process, a noisy sample s_t of an input object point cloud $s_0 = s$. Starting from a real LiDAR object s_0, the forward noising procedure produces a series of increasingly noisy 4D point clouds $s_1, \ldots, s_t$, finally arriving at $s_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I}) \in \mathbb{R}^4$. The training objective is to learn a distribution of the form $p_\theta(s_{t-1}|s_t) = \mathcal{N}(s_{t-1}|\boldsymbol{\mu}_\theta(s_t,t), \sigma_t^2\mathbf{I})$, parameterized by a denoising neural network.

The training loss optimizes the evidence variational lower bound, minimizing the Kullback-Leibler divergence between the forward and reverse processes. After simplification, it boils down to minimizing a mean-squared loss: $\arg\min_\theta ||\varepsilon_0 - \hat{\varepsilon}_\theta(x_t,t)||_2^2$ between the model output $\hat{\varepsilon}_\theta(s_t,t)$ and the ground-truth Gaussian noise ε_0. Given $\hat{\varepsilon}_\theta(s_t,t)$, we can then compute $\boldsymbol{\mu}_\theta(s_t,t)$ and get a denoising step recurrence equation defining s_{t-1} from s_t.

To exploit conditioning, the above formulation is extended with the extra input of a condition κ, leading to a denoising network $\hat{\varepsilon}_\theta(s_t,t,\kappa)$. To apply the conditional guidance, we follow the Classifer Free Guidance formulation from [13]. This implies the dropout of conditioning information during the training procedure, where all conditions are replaced by a pre-defined null embedding, which results in an unconditioned pass of the model.

3.2 Architecture

We now describe the architecture of LOGen and of our baselines, that we illustrate on Fig. 2. DiT-3D [25] is straightforwardly derived from DiT [30], and thus shares a similar architecture. PixArt-α [7] adds cross attention to DiT to inject conditions. Our baselines DiT-3DL and PixArt-L are directly adapted from their original counterparts to handle: (i) LiDAR data, i.e., point embedding, and (ii) LiDAR object conditioning. LOGen is then a further modification of PixArt-L, as described below. The conditioning $\kappa = (\phi, d, z, l, w, h)$ is applied in the denoising network to direct the generative process. Given the importance of the conditioning in our task, we explore three variants in the architecture design space.

- *AdaLN-Zero* is the best-performing guidance method proposed in DiT [30], and reused in DiT-3D. In our case, for DiT-3DL, the six conditioning values are concatenated to the time step and used to learn the adaptive layer-norm shift and scale parameters.

- *AdaLN-single* is the guidance proposed and used in PixArt-α [7]. While in the adaptive layer norm the condition is used to learn global shift and scale parameters, cross-attention layers allow the model to learn the guidance on a local scale. We do the same in PixArt-L.

- We observe that the separation of the self-attention layers from the cross-attention layers by a scale operation in the PixArt-L architecture degrades the generation results (cf. Tab. 1).

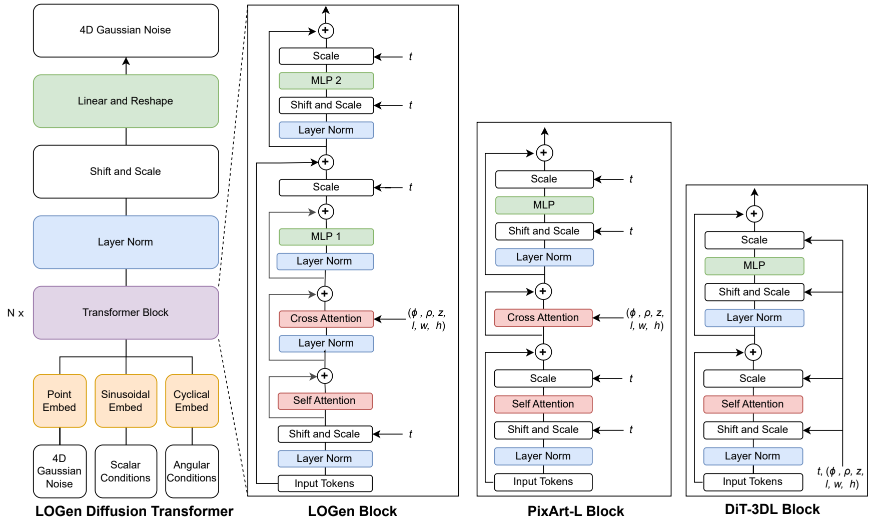

Figure 2: Architecture of LOGen and baselines PixArt-L and DiT-3DL.

Thus we introduce a third form of conditioning in LOGen blocks, where we apply the cross attention immediately after the self-attention layer, and scale after these operations.

4 Experiments

4.1 Experimental Setup

Datasets. We evaluate our approach on **nuScenes** [5], which was acquired with a 32-beam LiDAR, thus creating notably sparse sweep patterns. To construct object samples to learn from, we use the semantic point labels and the 3D box annotations of objects. We take as object instance all points falling into a box, filtering out possible points of a different class. We also discard objects with less than 20 points. Point intensity, which ranges in 0–255, has a highly irregular distribution, with most examples clustered from 0–30, and with outliers. To create a more uniform intensity distribution, we use log max scaling.

In total, we extract 268,684 objects with at least 20 points, which covers more than 95% of all object points. (Similarly, the official instance segmentation evaluation of nuScenes only considers instances with at least 15 points [5].) In order to unify the number of points per cloud in a batch, we pad each point cloud to the max length in the batch.

We further evaluate on objects extracted from **KITTI-360** [19]. We use semantic and instance labels from accumulated scans to extract roughly 40k objects in 3 classes as in [46]: *car*, *person*, *bicycle*. We also perform cross-dataset evaluation on **SemanticKITTI** [2].

For all experiments, we use the conditioning information from objects in the validation set, generating similar views of new objects of the same category.

Object-level metrics. To evaluate the ability to generate a sensible variant of a real object given only the conditioning information corresponding to the real object, we use two pointset metrics: Chamfer Distance (**CD**) and Earth Mover's Distance (**EMD**).

To evaluate perceptual realism, we use the Fréchet PointNet Distance (**FPD**) [35], an adaptation of the classic Fréchet Inception Distance (FID) [12], by replacing 2D Inception features with 3D PointNet features [31]. However, the original FPD used as reference relies on the original PointNet classifier, trained on ModelNet40 [31], which is not suited for LiDAR objects. We thus use instead a new PointNet classifier trained on the real LiDAR objects of nuScenes. We additionally consider the Kernel Inception Distance (KID), which has advantages over FID, in particular for small datasets [3]. Like for FID, we adapt KID into a Kernel PointNet Distance (**KPD**), replacing the Inception network by our trained PointNet.

We also consider distribution comparisons at a pointset level. We use coverage (**COV**) [1], which measures the proportion of real pointsets matched to at least one generated pointset, and 1-nearest neighbor accuracy (**1-NNA**) [47], derived from [22], which merges real and synthetic objects and counts the proportion of closest neighbors of each sample that are of the same source. 1-NNA ranges from 100% when distributions are separate, to 50% when they are indistinguishable. COV and 1-NNA are measured with CD and EMD. We extend COV and 1-NNA to measure the quality of the generated intensities. We extract 256-dimensional normalized intensity histograms from generated objects to use as features for COV and 1-NNA computations. Last, in [46], generated objects are extracted from generated scenes after being found by a 3D detector. On KITTI-360, we compare our generations to such detected objects, using object level metrics given in [46]. The Appendix details all metrics.

Scene-level metrics. A second set of evaluations is conducted on the perception of generated objects in a scene context by a typical 3D segmentation model with the contention that **more realistic objects should be segmented at a similar rate to real objects**. We consider the replacement of ground-truth objects by objects generated with the same conditioning, then measure the change in classwise **IoU** segmentation performance of the model on this modified data. We use a variant of nuScenes where each object with at least 20 points is replaced by a generated object of the same class and box size, seen under the same viewing angle, and with the same number of points. We highlight the correspondence between this metric and the #Box metric from [46], where both evaluate the expected performance at identifying generated objects of a model trained to identify real objects.

As segmenter, we use the popular SPVCNN [37], which has been widely used in evaluation settings [34, 39, 44]. We consider two different versions: **SPVCNN$_{\mathbf{3\,ch}}$** trained on points coordinates only, while **SPVCNN$_{\mathbf{4\,ch}}$** is also trained using point intensities.

Baselines. Due to task novelty, we build baselines: **MinkUNet** is the sparse convolutional architecture used in LiDiff [29], which ignores intensities. **PixArt-L** is our adaptation of PixArt-α to LiDAR points and conditioning, using our PointNet embeddings (PE) to allow intensity generation. **DiT-3DL VE$_{\mathbf{3\,ch}}$** is our adaptation of DiT-3D to LiDAR conditioning, i.e., with its original voxel embeddings (VE) and no intensity handling. (The voxelization library used in DiT-3D [25] does not handle a 4th input channel; handling intensities would require adapting the voxelization layer like in MinkUNet [8], as separate input features.) **DiT-3DL PE$_{\mathbf{4\,ch}}$** replaces VE by our PE in DiT-3DL VE$_{3\,\mathrm{ch}}$, thus also handling intensities.

4.2 Object-level evaluation

Qualitative generation examples are shown in Figs. 1, 3, 4 and the supplementary material.

Point embeddings. Table 1 shows that the voxel embeddings of DiT-3D do not perform as well as our PointNet embeddings, which supports our proposal (Sect. 3.2). MinkUNet suc-

Model	Emb.	Point Metrics CD	Point Metrics EMD	1-NNA (%) ↓ CD	1-NNA (%) ↓ EMD	1-NNA (%) ↓ Int.	COV (%) ↑ CD	COV (%) ↑ EMD	COV (%) ↑ Int.	FPD ↓ 3 ch.	FPD ↓ 4 ch.	KPD ↓ 3 ch.	KPD ↓ 4 ch.
MinkUNet	-	0.208	0.212	79.5	74.4	N/A	33.7	41.7	N/A	25.4	N/A	3.18	N/A
DiT-3DL	$\text{VE}_{3\,\text{ch}}$	0.136	0.121	75.8	70.9	N/A	34.3	42.0	N/A	2.80	N/A	0.25	N/A
DiT-3DL	$\text{PE}_{4\,\text{ch}}$	0.142	0.125	**73.4**	70.7	74.4	35.5	42.1	31.4	2.62	6.99	0.25	0.75
PixArt-L	$\text{PE}_{4\,\text{ch}}$	0.196	0.204	76.5	72.5	71.6	33.5	41.8	32.9	2.95	5.16	0.24	0.41
LOGen	$\text{PE}_{4\,\text{ch}}$	**0.130**	**0.111**	74.1	**70.6**	**68.2**	**36.0**	**42.3**	**37.0**	**1.34**	**2.18**	**0.10**	**0.12**

Table 1: **Combined point-space and feature-space evaluation.** CD and EMD are geometric fidelity metrics. 1-NNA and COV (in CD/EMD/intensity space) assess realism and coverage. FPD and KPD evaluate feature distribution alignment under 3- and 4-channel settings. Point embedding (Emb.) is: none (-), voxel-based ($\text{VE}_{3\,\text{ch}}$) or PointNet-based ($\text{PE}_{4\,\text{ch}}$).

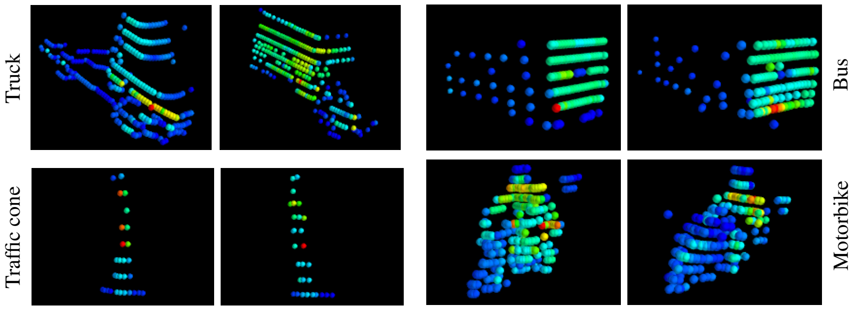

Figure 3: **Generations with intensities.** For each pair, the left object is from the nuScenes val set, the right object is generated using the same conditioning and number of points.

ceeds in generating local structure (LiDAR pattern), but fails to generate objects meeting size conditions. The lack of point embeddings in MinkUNet creates global size inconsistencies.

Architecture. Table 1 also shows that LOGen transformer blocks perform slightly better than the blocks proposed in PixArt-α and DiT-3D. The superiority of LOGen blocks is more pronounced as measured by the feature-based metrics of FPD and KPD.

Comparison on KITTI-360. The table in Fig. 4 shows the superior visual fidelity of objects generated by LOGen when compared to other scene and object generative methods.

4.3 Scene-level evaluation

Semantic quality of generated data. Table 2 shows that the LOGen generations are more semantically identifiable from the point of view of an SPVCNN segmenter. Though trained on real objects, the segmenter is able to recognize the semantics of LOGen objects at a much higher mIoU than other models, +19.6 pts vs the second best model. This is particularly pronounced for rare classes such as bicycle, bus, trailer, and construction vehicle. Besides, intensities generated by LOGen are realistic enough to provide +5.4 mIoU% at scene level.

4.4 Downstream evaluation

Object-level augmentation for semantic segmentation on nuScenes. To further motivate the focus on object level generation, we design a simple experiment highlighting one use case

Method	CD↓	JSD↓
LiDARGen [51]	33.10	0.41
UltraLiDAR [45]	18.30	0.70
R2DM [27]	17.60	0.86
OLiDM [46]	1.88	0.91
LOGen (ours)	**0.24**	**0.19**

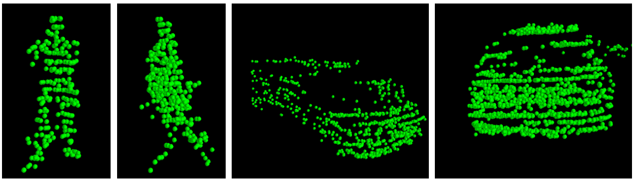

Figure 4: **Quantitative & qualitative comparison of LiDAR objects on KITTI-360 [19].** (Left) Evaluation on 1000 sampled objects (for LOGen) or detected objects (for others) using Chamfer Distance (CD) and Jensen-Shannon Divergence (JSD). (Right) Examples of generated objects from KITTI-360 for *car* and *person*. For classwise results of LOGen, along with additional metrics EMD, FPD, and KPD, please refer to the supplementary material.

	Model	Emb.	barrier	bicycle	bus	car	con.veh.	moto.	pedes.	tra.con.	trailer	truck	mean
3 channels	MinkUNet	-	76.1	31.4	18.7	**84.4**	14.7	63.6	69.8	59.3	12.4	49.7	48.0
	DiT-3DL	VE 3 ch	48.4	42.5	0.3	34.1	0.2	63.8	75.6	69.6	7.7	3.4	34.6
	DiT-3DL	PE 4 ch	61.6	39.2	11.1	84.2	1.7	68.8	77.5	51.3	15.9	26.6	43.9
	PixArt-L	PE 4 ch	58.9	32.1	3.9	75.7	0.9	76.6	79.0	56.7	13.6	51.3	44.9
	LOGen	PE 4 ch	**78.2**	**50.9**	**60.1**	79.1	**28.1**	**82.2**	**84.2**	**73.5**	**37.5**	**63.2**	**63.7**
	GT		80.3	48.8	90.7	94.2	40.6	86.0	85.5	73.4	60.7	85.2	74.5
4 channels	DiT-3DL	PE 4 ch	62.5	29.0	18.3	85.9	6.3	62.6	68.4	47.2	15.9	37.7	43.4
	PixArt-L	PE 4 ch	56.6	33.7	14.5	82.4	8.1	78.5	79.3	62.8	26.7	52.6	49.5
	LOGen	PE 4 ch	**78.0**	**48.9**	**71.7**	**91.3**	**37.1**	**82.0**	**84.8**	**79.1**	**45.1**	**72.9**	**69.1**
	GT		81.6	50.4	90.9	94.6	44.5	88.1	85.6	77.4	64.6	85.6	76.3

Table 2: **Scene-level evaluation.** SPVCNN is trained with 3 or 4 channels (coordinates + intensity) on nuScenes (train set) with real objects. It is tested on nuScenes (val set) where each object (with ≥ 20 points) is replaced with a generated object of the same class, box size and sensor viewing angle, and with the same number of points, and where objects with less than 20 points are removed from the dataset.

of the generated objects, enhancing diversity of objects in the training dataset. We train the SPVCNN [37] model for semantic segmentation on the nuScenes dataset. During training, with a 50% probability, we replace real objects with versions generated by LOGen. Results are in Tab. 3. We highlight the improvements for specific rare classes such as traffic cones, trailers, and trucks, and further improvements for cars. Note that, in this downstream task, using a single-class model is consistent with the selection of specific rare classes to augment.

Cross-dataset object augmentation. We use LOGen trained on nuScenes to enhance the semantic segmentation performance on SemanticKITTI for classes *pedestrian* and *motorcycle*, which are commonly targeted for augmentation due to the relatively low performance of their segmentation. We randomly insert 10 such generated objects for each class into SemanticKITTI scenes during SPVCNN training. This simple augmentation strategy improves classwise IoU% from 64.8 to 67.3 for *motorcycle* and from 65.0 to 66.9 for *pedestrian*.

We provide a comprehensive supplementary material that includes detailed information on the diffusion background, implementation specifics, official metric definitions, dataset statistics, evaluation of conditioning parameters, all classwise performance results, comparisons between LOGen's single-model-per-class and multi-class approaches, qualitative comparisons between LOGen and baselines, examples of novel LOGen objects and videos showcasing our object generation results.

Mean	barrier	bicycle	bus	car	constr. veh.	motorcycle	pedestrian	traf. cone	trailer	truck
+**0.3**	−0.5	−0.8	−1.2	+0.7	−1.7	+0.5	0	+4.3	+0.3	+1.1

Table 3: **Impact of inplace data augmentation with LOGen samples for semantic segmentation on nuScenes validation set.** During training, each sample is selected with a 50% probability to be either a real sample or a generated one. Results are classwise IoU%.

LOGen PE $_{4\,ch}$	CD ↓	EMD ↓	1-NNA ↓ *CD*	1-NNA ↓ *EMD*	COV ↑ *CD*	COV ↑ *EMD*	FPD ↓ 3 ch.	FPD ↓ 4 ch.	KPD ↓ 3 ch.	KPD ↓ 4 ch.	APC ↑ 3 ch.	APC ↑ 4 ch.
Single-model-per-class	**0.130**	**0.111**	74.1	70.6	36.0	42.3	**1.34**	**2.18**	**0.10**	**0.12**	**40.9**	**48.1**
Multi-class model	0.149	0.134	**72.8**	**68.9**	**37.1**	**43.3**	2.35	4.28	0.21	0.45	36.2	42.5

Table 4: **Single-model-per-class vs. multi-class model.** Comparison of a same size single-model-per-class and multi-class models on object generation metrics. The multi-class model lags behind the single-model-per-class approach in realism metrics.

4.5 A discussion on scalability

Following other works in the object generation literature [25], we mainly chose to train and evaluate a single model per class. This allowed us to use a compact network, and avoid inconsistent batch sizes. Due to the use of zero-padding, mixing objects with high variation in size results in certain objects being padded with many zero elements. We note that training single model per class introduces no drawbacks in terms of runtime or model efficiency when evaluated in iterations per instance.

We train and evaluate a multi-class model of the same size trained with the same iterations as shown in Table 4, where the multi-class model does not perform as well as the single class versions under most perceptual metrics. However the multi-class model does have the best overall 1-NNA and Coverage, indicating avenues for future work.

Further, while the point embeddings outperform the voxelization approaches in terms of generated quality, the lack of any point compression creates memory consumption issues for larger objects.

5 Conclusion

This work demonstrates that a well-designed diffusion model is capable of generating realistic LiDAR objects. Novel metrics, baselines and LOGen (best performing architecture) establish the first standard in LiDAR object generation. Interesting perspectives include further development of downstream applications. Future work can increase generation speed and memory efficiency of the models, and develop novel methods to use generated objects.

Acknowledgments

This work was granted access to the HPC resources of IDRIS under the allocations AD011013503R1 and AD011013503R3 made by GENCI.

References

[1] Panos Achlioptas, Olga Diamanti, Ioannis Mitliagkas, and Leonidas Guibas. Learning representations and generative models for 3D point clouds. In *ICML*, 2018.

[2] Jens Behley, Martin Garbade, Andres Milioto, Jan Quenzel, Sven Behnke, Cyrill Stachniss, and Jurgen Gall. SemanticKITTI: A Dataset for Semantic Scene Understanding of LiDAR Sequences. In *ICCV*, 2019.

[3] Mikolaj Binkowski, Danica J. Sutherland, Michael Arbel, and Arthur Gretton. Demystifying MMD GANs. In *ICLR*, 2018.

[4] Alexandru Buburuzan, Anuj Sharma, John Redford, Puneet K Dokania, and Romain Mueller. Mobi: Multimodal object inpainting using diffusion models. *arXiv:2501.03173*, 2025. CVPR 2025 Workshop on Data-Driven Autonomous Driving Simulation (DDADS).

[5] Holger Caesar, Varun Bankiti, Alex H Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan, Giancarlo Baldan, and Oscar Beijbom. nuScenes: A multimodal dataset for autonomous driving. In *CVPR*, pages 11621–11631, 2020.

[6] Angel X. Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, Jianxiong Xiao, Li Yi, and Fisher Yu. ShapeNet: An information-rich 3D model repository. Technical Report arXiv:1512.03012, Stanford University & Princeton University & Toyota Technological Institute at Chicago, 2015.

[7] Junsong Chen, Jincheng Yu, Chongjian Ge, Lewei Yao, Enze Xie, Yue Wu, Zhongdao Wang, James Kwok, Ping Luo, Huchuan Lu, et al. Pixart-α: Fast training of diffusion transformer for photorealistic text-to-image synthesis. In *ICLR*, 2024.

[8] Christopher Choy, JunYoung Gwak, and Silvio Savarese. 4D spatio-temporal ConvNets: Minkowski convolutional neural networks. In *CVPR*, 2019.

[9] Jin Fang, Dingfu Zhou, Feilong Yan, Tongtong Zhao, Feihu Zhang, Yu Ma, Liang Wang, and Ruigang Yang. Augmented lidar simulator for autonomous driving. *IEEE Robotics and Automation Letters*, 5(2):1931–1938, 2020.

[10] Jin Fang, Xinxin Zuo, Dingfu Zhou, Shengze Jin, Sen Wang, and Liangjun Zhang. LiDAR-Aug: A general rendering-based augmentation framework for 3D object detection. In *CVPR*, 2021.

[11] Michael Gschwandtner, Roland Kwitt, Andreas Uhl, and Wolfgang Pree. BlenSor: Blender sensor simulation toolbox. In *Advances in Visual Computing (ISVC)*, 2011.

[12] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. GANs trained by a two time-scale update rule converge to a local Nash equilibrium. In *NeurIPS*, 2017.

[13] Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance. *arXiv preprint arXiv:2207.12598*, 2022. Also in NeurIPS 2021 Workshop on Deep Generative Models and Downstream Applications.

[14] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In *NeurIPS*, 2020.

[15] Qianjiang Hu, Zhimin Zhang, and Wei Hu. RangeLDM: Fast realistic lidar point cloud generation. In *ECCV*, 2024.

[16] Haebom Lee, Christian Homeyer, Robert Herzog, Jan Rexilius, and Carsten Rother. Spatio-temporal outdoor lighting aggregation on image sequences using transformer networks. *IJCV*, 131(4):1060–1072, 2023.

[17] Kaican Li, Kai Chen, Haoyu Wang, Lanqing Hong, Chaoqiang Ye, Jianhua Han, Yukuai Chen, Wei Zhang, Chunjing Xu, Dit-Yan Yeung, et al. Coda: A real-world road corner case dataset for object detection in autonomous driving. In *ECCV*, 2022.

[18] Shidi Li, Miaomiao Liu, and Christian Walder. EditVAE: Unsupervised parts-aware controllable 3D point cloud shape generation. In *AAAI*, 2022.

[19] Yiyi Liao, Jun Xie, and Andreas Geiger. Kitti-360: A novel dataset and benchmarks for urban scene understanding in 2d and 3d. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(3):3292–3310, 2022.

[20] Mingyu Liu, Ekim Yurtsever, Jonathan Fossaert, Xingcheng Zhou, Walter Zimmer, Yuning Cui, Bare Luka Zagar, and Alois C Knoll. A survey on autonomous driving datasets: Statistics, annotation quality, and a future outlook. *IEEE Transactions on Intelligent Vehicles*, 2024.

[21] Thibaut Loiseau, Tuan-Hung Vu, Mickael Chen, Patrick Pérez, and Matthieu Cord. Reliability in semantic segmentation: Can we use synthetic data? In *ECCV*, pages 442–459. Springer, 2024.

[22] David Lopez-Paz and Maxime Oquab. Revisiting classifier two-sample tests. In *ICLR*, 2018.

[23] Shitong Luo and Wei Hu. Diffusion probabilistic models for 3D point cloud generation. In *CVPR*, 2021.

[24] Sivabalan Manivasagam, Shenlong Wang, Kelvin Wong, Wenyuan Zeng, Mikita Sazanovich, Shuhan Tan, Bin Yang, Wei-Chiu Ma, and Raquel Urtasun. LiDARsim: Realistic lidar simulation by leveraging the real world. In *CVPR*, 2020.

[25] Shentong Mo, Enze Xie, Ruihang Chu, Lanqing Hong, Matthias Niessner, and Zhenguo Li. DIT-3D: Exploring plain diffusion transformers for 3D shape generation. In *NeurIPS*, 2023.

[26] Khan Muhammad, Tanveer Hussain, Hayat Ullah, Javier Del Ser, Mahdi Rezaei, Neeraj Kumar, Mohammad Hijji, Paolo Bellavista, and Victor Hugo C de Albuquerque. Vision-based semantic segmentation in scene understanding for autonomous driving: Recent achievements, challenges, and outlooks. *IEEE Transactions on Intelligent Transportation Systems*, 23(12):22694–22715, 2022.

[27] Kazuto Nakashima and Ryo Kurazume. Lidar data synthesis with denoising diffusion probabilistic models. In *ICML*, 2024.

[28] Michael Niemeyer and Andreas Geiger. Giraffe: Representing scenes as compositional generative neural feature fields. In *CVPR*, pages 11453–11464, 2021.

[29] Lucas Nunes, Rodrigo Marcuzzi, Benedikt Mersch, Jens Behley, and Cyrill Stachniss. Scaling diffusion models to real-world 3D lidar scene completion. In *CVPR*, 2024.

[30] William Peebles and Saining Xie. Scalable diffusion models with transformers. In *ICCV*, 2023.

[31] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. PointNet: Deep learning on point sets for 3D classification and segmentation. In *CVPR*, 2017.

[32] Haoxi Ran, Vitor Guizilini, and Yue Wang. Towards realistic scene generation with lidar diffusion models. In *CVPR*, 2024.

[33] Jeremy Reizenstein, Roman Shapovalov, Philipp Henzler, Luca Sbordone, Patrick Labatut, and David Novotny. Common objects in 3d: Large-scale learning and evaluation of real-life 3d category reconstruction. In *ICCV*, 2021.

[34] Nermin Samet, Oriane Siméoni, Gilles Puy, Georgy Ponimatkin, Renaud Marlet, and Vincent Lepetit. You never get a second chance to make a good first impression: Seeding active learning for 3D semantic segmentation. In *ICCV*, 2023.

[35] Dong Wook Shu, Sung Woo Park, and Junseok Kwon. 3D point cloud generative adversarial network based on tree structured graph convolutions. In *ICCV*, 2019.

[36] Matthew Tancik, Pratul P. Srinivasan, Ben Mildenhall, Sara Fridovich-Keil, Nithin Raghavan, Utkarsh Singhal, Ravi Ramamoorthi, Jonathan T. Barron, and Ren Ng. Fourier features let networks learn high frequency functions in low dimensional domains. In *NeurIPS*, 2020.

[37] Haotian Tang, Zhijian Liu, Shengyu Zhao, Yujun Lin, Ji Lin, Hanrui Wang, and Song Han. Searching efficient 3d architectures with sparse point-voxel convolution. In *ECCV*, 2020.

[38] Michał J Tyszkiewicz, Pascal Fua, and Eduard Trulls. GECCO: Geometrically-conditioned point diffusion models. In *ICCV*, 2023.

[39] Ozan Unal, Dengxin Dai, Ali Tamer Unal, and Luc Van Gool. Discwise active learning for lidar semantic segmentation. *IEEE Robotics and Automation Letters (RA-L)*, 8(11): 7671–7678, 2023.

[40] Arash Vahdat, Francis Williams, Zan Gojcic, Or Litany, Sanja Fidler, Karsten Kreis, et al. LION: Latent point diffusion models for 3D shape generation. In *NeurIPS*, 2022.

[41] Kasi Viswanath, Peng Jiang, and Srikanth Saripalli. Reflectivity is all you need!: Advancing lidar semantic segmentation. In *IFAC Symposium on Intelligent Autonomous Vehicles (IAV)*, 2025.

[42] Yang Wu, Kaihua Zhang, Jianjun Qian, Jin Xie, and Jian Yang. Text2lidar: Text-guided lidar point cloud generation via equirectangular transformer. In *ECCV*, pages 291–310. Springer, 2024.

[43] Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao. 3D ShapeNets: A deep representation for volumetric shapes. In *CVPR*, 2015.

[44] Binhui Xie, Shuang Li, Qingju Guo, Chi Harold Liu, and Xinjing Cheng. Annotator: An generic active learning baseline for lidar semantic segmentation. In *NeurIPS*, 2023.

[45] Yuwen Xiong, Wei-Chiu Ma, Jingkang Wang, and Raquel Urtasun. UltraLiDAR: Learning compact representations for lidar completion and generation. In *CVPR*, 2023.

[46] Tianyi Yan, Junbo Yin, Xianpeng Lang, Ruigang Yang, Cheng-Zhong Xu, and Jianbing Shen. Olidm: Object-aware lidar diffusion models for autonomous driving. In *AAAI*, 2025.

[47] Guandao Yang, Xun Huang, Zekun Hao, Ming-Yu Liu, Serge Belongie, and Bharath Hariharan. PointFlow: 3D point cloud generation with continuous normalizing flows. In *ICCV*, 2019.

[48] Xiangyu Yue, Bichen Wu, Sanjit A. Seshia, Kurt Keutzer, and Alberto L. Sangiovanni-Vincentelli. A lidar point cloud generator: from a virtual world to autonomous driving. In *ACM International Conference on Multimedia Retrieval (ICMR)*, 2018.

[49] Jinlai Zhang, Lyujie Chen, Bo Ouyang, Binbin Liu, Jihong Zhu, Yujin Chen, Yanmei Meng, and Danfeng Wu. PointCutMix: Regularization strategy for point cloud classification. *Neurocomputing*, 505:58–67, 2022.

[50] Junge Zhang, Feihu Zhang, Shaochen Kuang, and Li Zhang. NeRF-LiDAR: Generating realistic lidar point clouds with neural radiance fields. In *AAAI*, 2024.

[51] Vlas Zyrianov, Xiyue Zhu, and Shenlong Wang. Learning to generate realistic lidar point clouds. In *ECCV*, 2022.

[52] Vlas Zyrianov, Henry Che, Zhijian Liu, and Shenlong Wang. LidarDM: Generative lidar simulation in a generated world. In *ICRA*, 2025.