DiffuBox: Refining 3D Object Detection with Point Diffusion

Xiangyu Chen[*1,†] **Zhenzhen Liu**[*1] **Katie Z Luo**[*1] **Siddhartha Datta**[2]
Adhitya Polavaram[1] **Yan Wang**[3] **Yurong You**[3] **Boyi Li**[3] **Marco Pavone**[3]
Wei-Lun Chao[4] **Mark Campbell**[1] **Bharath Hariharan**[1] **Kilian Q. Weinberger**[1]
[1]Cornell University [2]University of Oxford [3]NVIDIA Research [4]The Ohio State University

Abstract

Ensuring robust 3D object detection and localization is crucial for many applications in robotics and autonomous driving. Recent models, however, face difficulties in maintaining high performance when applied to domains with differing sensor setups or geographic locations, often resulting in poor localization accuracy due to domain shift. To overcome this challenge, we introduce a novel diffusion-based box refinement approach. This method employs a domain-agnostic diffusion model, conditioned on the LiDAR points surrounding a coarse bounding box, to simultaneously refine the box's location, size, and orientation. We evaluate this approach under various domain adaptation settings, and our results reveal significant improvements across different datasets, object classes and detectors. Our PyTorch implementation is available at https://github.com/cxy1997/DiffuBox.

1 Introduction

3D object detection is a fundamental task for embodied agents to safely navigate in complex environments. For autonomous vehicles to navigate complicated traffic conditions, this amounts to identifying and localizing other road agents. Detection models under this setting need to make sense of LiDAR point clouds to identify accurate bounding boxes for pre-specified objects. Given the diverse driving environments that occur in practice it is common, however, for the train- and test-time distributions to differ significantly. Domain distributional differences mainly arise from differences in object size, point cloud density, and LiDAR beam angles. Consequently, models trained in one region or particular dataset (*e.g*. Germany) may not perform well in another region or dataset (*e.g*. USA) [48]. As a result, the domain adaptation problem raises concerns over the reliability and safety of 3D object detection in self-driving, that are often trained in a particular setting, then deployed into a diverse set of regions and locations.

Wang *et al*. [48] have obtained reductions in the domain adaptation gap by resizing boxes with a simple scaling heuristic after the fact. Consequently, we share the belief that the performance gap associated with domain adaptation is dominated by incorrect box sizes, shapes, and orientations, rather than false positives and negatives in detections — *e.g*. a model trained in Germany can detect US cars, but struggles to capture their larger dimensions.

In this paper we observe that, although the relationship of bounding boxes to the surrounding environment varies across domains, the relative position of LiDAR points with respect to their bounding boxes is surprisingly consistent [23]. Bounding boxes of these detections are, by definition, supposed to tightly fit the corresponding objects. Furthermore, the objects within the same object class (*e.g*., cars) have a similar shape with minor variance across different domains. What mostly

[*]Denotes equal contribution.
[†]Correspondences could be directed to `xc429@cornell.edu`

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

 https://doi.org/10.52202/079017-3293

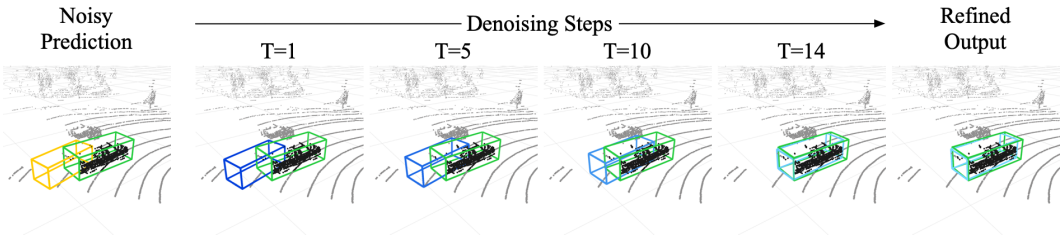


Figure 1: **Box refinement through denoising steps.** We visualize the correction of a noisy prediction, shown in yellow, using DiffuBox. The ground truth box is visualized in green for reference. Boxes being refined are colored blue based on timestep. The output is refined iteratively though the denoising steps, resulting in the final, corrected output of our method.

varies, then, is the dimensions of the object, as opposed to this "surface" shape when normalized to be the same size. Specifically, the distribution of points the LiDAR detector receives, is therefore consistent when normalized across object sizes, regardless of domains; points will always land near the edge of the bounding box no matter where the object is located. Thus, if we can somehow capture the distribution of points relative to a box's coordinate system, we would be able to use same process to fix incorrectly positioned bounding boxes to fit the correct point-distribution, even across domains.

Recognizing this observation, we propose DiffuBox, a novel point diffusion model that learns the distribution of points relative to the object's bounding box in order to refine noisy bounding box proposals from the detection models for off-the-shelf domain adaptation. Given a set of noisy bounding box proposals, DiffuBox denoises them into accurate detection boxes conditioned on the points near proposed bounding boxes. Our method naturally avoids the domain gap caused by the scale difference [48], since DiffuBox is designed to operate on object scale-invariant data, where we transform the LiDAR points around bounding box proposals into a normalized box view that is relative to the box instead of in absolute measure. This eliminates the size priors presented in the source domain and forces the diffusion model to recover the accurate bounding box solely based on the relative position of points to the bounding box proposal, allowing for improved robustness in self-driving systems.

To summarize, our contributions include: We empirically validate our method, DiffuBox, by adapting models trained on a dataset from Germany (KITTI [7]) into two large, real-world datasets from the USA (Lyft L5 [15] and Ithaca365 [5]). Under both settings, we observe that DiffuBox is able to refine the output bounding boxes drastically from the noisy initial predictions (Figure 1). Quantitatively, we observe strong improvements in mAP performance (up to 24 mAP), particularly in near-range boxes, where more points are present for DiffuBox to refine the box predictions. When paired with a representative set of domain adaptation methods, including Output Transform, Statistical Normalization [48], and Rote-Domain Adaptation [59], DiffuBox is able to further improve the results, and closing the gap between all method's final performance.

2 Related Work

3D Object Detection. In general, most 3D object detection methods require supervision from human-annotated data. They take 3D sensory data (e.g. LiDAR point clouds) and infer bounding boxes around 3D objects. 3D detection methods can be grouped into two categories based on the input representations: Point-based methods [32, 33, 31, 39, 40, 55, 29] that directly operate on point clouds, and grid-based methods [51, 65, 20, 38, 49, 26] that first voxelize point clouds into 3D grids and then leverage convolutional architectures. Like other supervised models, 3D detection models suffer from decreased performance when the data distribution during inference differs from that during training. Our method DiffuBox is designed to reduce the domain gap for general 3D object detection, agnostic to underlining model design.

Domain Adaptation in 3D. Domain adaptation aims to alleviate the performance drop of 3D perception models under domain shift. [48] by Wang *et al*. is one of the first works studying the domain gap in 3D object detection and proposes Statistical Normalization (SN) that reduces the shape bias across domains. ST3D [53], Rote-DA [59], and ST3D++ [54] propose a self-training pipeline that iteratively improve the target domain 3D detection performance with pseudo-label training

and auxiliary priors. Other methods can be grouped into feature-based [50, 21, 28, 19, 37, 17] and architecture-based [12, 35, 46, 22, 36] methods. Some of them also apply data augmentation to construct and train domain-invariant representations to reduce the domain gap [19, 37, 12]. Our proposed method is orthogonal to these methods and can be applied together with these models.

Diffusion Models. Recently, diffusion models [41, 9, 42, 43] have shown high-quality generative ability for image [4, 34], video [10, 8] and 3D shape [25, 61, 27] modalities. Zhou *et al*. [62] uses diffusion models with a point-voxel representation for shape generation and point-cloud completion. LION [61] uses a hierarchical VAE mapped to a latent space and trains diffusion models on latent encodings to generate point clouds. In perception tasks, Chen *et al*. [3] and Zhou *et al*. [64] propose diffusion-based object detection frameworks. Kim *et al*. [18] proposes a diffusion-based module to enhance the proposal refinement stage of two-stage object detectors. Unlike these approaches, our work focuses on leveraging diffusion for post-processing in a detector-agnostic manner that shows superior performance over previous methods.

3 Method

3.1 Problem Setup

Despite great in-domain performance, 3D object detection models often struggle to maintain their accuracy when generalized to new domains (datasets). It has been concluded that such poor performance is mainly caused by mislocalization rather than misdetection [48]. That is, although objects can be correctly recognized by the object detector, the detected boxes lack sufficient overlap with the ground truth box and do not count as true positive (*i.e*., detections with IoU < 0.7 with ground-truth).

In this work we introduce DiffuBox, which focuses on correcting the localization of bounding box proposals, as illustrated in Figure 3, to improve domain adaptation for 3D object detection. Unlike existing domain adaptation algorithms that require careful re-training on the target [59, 53] or source [48] domain data, DiffuBox can be deployed off-the-shelf as a post-processing procedure in any novel domain.

Let $\boldsymbol{P} \in \mathcal{R}^{N\times 3}$ denote a N-point 3D point cloud from the target domain. Let $\mathcal{B} = \{\boldsymbol{b}_1, \ldots, \boldsymbol{b}_M\}$ be a set of M imperfect bounding boxes proposed by an underadapted object detector given $\boldsymbol{P}$, where each bounding box $\boldsymbol{b}_i$ is a 7-DoF (degrees of freedom) upright box, parameterized with center $[x_i, y_i, z_i]$, size $[w_i, l_i, h_i]$ and yaw angle θ_i. We aim to obtain better localized object proposals $\hat{\mathcal{B}}$ by refining the boxes in $\mathcal{B}$ *without any re-training*.

$$\hat{\mathcal{B}} = \{\hat{\boldsymbol{b}}_1, \ldots, \hat{\boldsymbol{b}}_M \mid \hat{\boldsymbol{b}}_i = \text{refine}(\boldsymbol{b}_i, \boldsymbol{P})\}. \tag{1}$$

3.2 Learning Shapes in the Normalized Box View

While domain differences between 3D object detection datasets exist in many aspects, the analysis from [48] shows that the most significant hurdle for adaptation comes from the difference in object size. For instance, the American cars in the Lyft dataset [15] are about 20% larger than German cars in the KITTI dataset [7] on average, and an object detector trained on KITTI will tend to still predict small boxes when tested on Lyft. Unfortunately, as long as 3D object detectors are trained to explicitly predict object sizes, such size priors will be inevitably memorized during training and carried on to other domains as learned bias.

We aim to achieve **scale-invariant object detection**, which would be naturally immune to size priors. Motivated by Luo *et al*.'s [23] finding that the relative distribution of points to ground-truth bounding boxes is consistent across domains, *i.e*. points tend to concentrate near the surface of boxes, we propose to disentangle object size from shape by transforming pointclouds into a normalized box view (NBV), where point coordinates are box-relative rather than absolute.

Using homogeneous transformation, we define $\boldsymbol{P}_{\boldsymbol{b}}^{\text{NBV}} \in \mathcal{R}^{N\times 3}$, the normalized box view of point cloud $\boldsymbol{P}$ relative to bounding box $\boldsymbol{b} = [x, y, z, w, l, h, \theta]$, to be

$$\begin{bmatrix} \boldsymbol{P}_{\boldsymbol{b}}^{\text{NBV}} \\ \mathbf{1} \end{bmatrix} = \begin{bmatrix} \frac{2}{l} & 0 & 0 & 0 \\ 0 & \frac{2}{w} & 0 & 0 \\ 0 & 0 & \frac{2}{h} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\theta & \sin\theta & 0 & 0 \\ -\sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & -x \\ 0 & 1 & 0 & -y \\ 0 & 0 & 1 & -z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \boldsymbol{P} \\ \mathbf{1} \end{bmatrix} \tag{2}$$

https://doi.org/10.52202/079017-3293

Figure 2: Example *Car* objects converted into normalized box view (NBV). Foreground/background points are marked in black/gray, respectively for better visualization. Foreground LiDAR points distributing tightly within a $[-1, 1]^3$ NBV cube is a domain-consistent sign for good localization.

As shown in Figure 2, Equation 2 transforms the bounding box $\boldsymbol{b}$ into a $[-1, 1]^3$ cube, eliminating the size prior. The same transform also transforms $\boldsymbol{P}$ into box-relative, scale-invariant $\boldsymbol{P}_{\boldsymbol{b}}^{\text{NBV}}$.

In practice, we only consider points within a certain depth range of the bounding box for efficiency. We refer to this range as *context limit*. In the sections below, we overload $\boldsymbol{P}_{\boldsymbol{b}}^{\text{NBV}}$ as the point cloud within the context limit for ease of reference.

3.3 Bounding Box Refinement via Diffusion

Inspired by recent works on diffusion-based shape generation [52, 63, 25, 61] and knowledge distillation from pretrained diffusion models [30], we show that size-agnostic shape knowledge learned by a point cloud diffusion model can help to improve object localization across domains. The underlining assumption is that despite size difference, objects of the same category (*e.g. car*, *cyclist*, *pedestrian*) share similar shapes.

Figure 2 illustrates our hypothesis that the good localization of a bounding box $\boldsymbol{b}$ is closely correlated to its corresponding $\boldsymbol{P}_{\boldsymbol{b}}^{\text{NBV}}$ forming a "standard" point distribution, *a.k.a.* shape. Thus, improving the shape of $\boldsymbol{P}_{\boldsymbol{b}}^{\text{NBV}}$ will also lead to better localization of $\boldsymbol{b}$. Since our ultimate goal is to optimize the bounding box $\boldsymbol{b} \rightarrow \hat{\boldsymbol{b}}$, we propose to use a point diffusion model to learn to "denoise" $\boldsymbol{P}_{\boldsymbol{b}}^{\text{NBV}} \rightarrow \boldsymbol{P}_{\hat{\boldsymbol{b}}}^{\text{NBV}}$.

Specifically, we begin by discussing the training of the diffusion model to learn the probabilistic flow for each point to a good box in Section 3.3.1. Then, we go into how our method refines the bounding box by computing the improvement step relative to the learned probabilistic flow in Section 3.3.2. Finally, we go into how we leverage the shape guidance to embed heuristics into our training procedure in Section 3.3.3. To enhance clarity, we additionally provide an algorithmic description of DiffuBox's training and inference workflow in Appendix S1.

3.3.1 Diffusion Training

The learning objective of a diffusion model can be viewed as a variant of the score function [43] $\nabla_{\boldsymbol{x}} \log p(\boldsymbol{x}; \sigma)$, where σ indicates the noise level and $p(\boldsymbol{x}; 0) = p_{\text{data}}$, the true data distribution that is hard to directly sample from. As the score function points data to a higher likelihood, samples can instead be drawn from $p(\boldsymbol{x}; \sigma_{\text{max}})$ — which is usually modeled as an i.i.d. Gaussian distribution — and denoised into p_{data} by solving a probabilistic flow ODE [11]/SDE [45].

Let $F_{\boldsymbol{\theta}}$ denote a diffusion model with parameter $\boldsymbol{\theta}$. Considering the full design space [13], in general its training loss can be written as:

$$\mathbb{E}_{\sigma, \boldsymbol{y}, \boldsymbol{n}} \left[\lambda(\sigma) c_{\text{out}}(\sigma)^2 \left\| F_\theta \left(c_{\text{in}}(\sigma)(\boldsymbol{y} + \boldsymbol{n}); c_{\text{noise}}(\sigma) \right) - \frac{1}{c_{\text{out}}(\sigma)} \left(\boldsymbol{y} - c_{\text{skip}}(\sigma)(\boldsymbol{y} + \boldsymbol{n}) \right) \right\|_2^2 \right], \quad (3)$$

where $\sigma \sim p_{\text{train}}$, $\boldsymbol{y} \sim p_{\text{data}}$, and $\boldsymbol{n} \sim \mathcal{N}(0, \sigma^2 \boldsymbol{I})$. $\lambda(\sigma)$ denotes the effective training weight, $c_{\text{noise}}(\sigma)$ denotes noise level preconditioning. $c_{\text{in}}(\sigma)$, $c_{\text{out}}(\sigma)$, and $c_{\text{skip}}(\sigma)$ are input/output scaling factors.

Modelling $p(\boldsymbol{x}; \sigma_{\text{max}})$ as $\mathcal{N}(0, \boldsymbol{I})$ allows for easy sampling. However, an i.i.d. Gaussian noise doesn't suit the shift of point cloud in NBV caused by object mislocalization. As shown in Figure S8,

a noisy NBV point cloud is formed from a 3D distortion on the standard shape, rather than adding Gaussian noise. The distortion includes rotation (caused by incorrect raw angle), rescaling (caused by incorrect size), and translation (caused by incorrect box center).

Because of this, we made a few adaptations to the diffusion process. We set the effective training weight $\lambda(\sigma) = 1$, all input/output scaling factors $c_{\text{in}}(\sigma) = c_{\text{out}}(\sigma) = c_{\text{skip}}(\sigma) = 1$, as the noise level σ is unknown during inference. We apply Gaussian noise on the bounding box, rather than on the point cloud, to simulate mislocalized bounding boxes. With these adapatations, our new training loss becomes:

$$\mathbb{E}_{\sigma,(\boldsymbol{P},\boldsymbol{b}^*),\boldsymbol{n}}\left[\lambda(\sigma)\|F_\theta\left(\boldsymbol{P}^{\text{NBV}}_{\boldsymbol{b}^*+\boldsymbol{n}}; c_{\text{noise}}(\sigma)\right) - \left(\boldsymbol{P}^{\text{NBV}}_{\boldsymbol{b}^*} - \boldsymbol{P}^{\text{NBV}}_{\boldsymbol{b}^*+\boldsymbol{n}}\right)\|_2^2\right] \tag{4}$$

where $\sigma \sim p_{\text{train}}$, $\boldsymbol{P}, \boldsymbol{b}^* \sim \mathcal{D}_{\text{train}}$ and $\boldsymbol{n} \sim \mathcal{N}\left(0, \sigma^2 diag\left(\boldsymbol{\Sigma}\right)\right)$. $diag(\boldsymbol{\Sigma})$ is the variance of box noise, which is roughly estimated from *direct* domain adaptation performance.

3.3.2 Bounding Box Updates

Since $F_{\boldsymbol{\theta}}$ is trained to approximate the score function $\nabla_{\boldsymbol{x}} \log p\left(\boldsymbol{x}; \sigma\right)$, the regular denoising process can be implemented by solving a probablistic flow ODE:

$$\mathrm{d}\boldsymbol{x} = -\dot{\sigma}(t)\sigma(t)\nabla_{\boldsymbol{x}} \log p\left(\boldsymbol{x}; \sigma(t)\right) \mathrm{d}t, \tag{5}$$

where $\sigma(t)$ denotes a noise schedule in which $\sigma(T) = \sigma_{\text{max}}$, $\sigma(0) = 0$, and the dot stands for a time derivative. Thus, $\boldsymbol{x}_0 \sim p_{\text{data}}$ can be generated by evolving $\boldsymbol{x}_T \sim p\left(\boldsymbol{x}; \sigma_{\text{max}}\right) = \mathcal{N}\left(0; \sigma^2 \boldsymbol{I}\right)$ from time $t = T$ to $t = 0$.

The NBV point cloud diffusion model $F_\theta\left(\boldsymbol{P}^{\text{NBV}}_{\boldsymbol{b}}; c_{\text{noise}}(\sigma)\right) \approx \frac{\nabla \log p\left(\boldsymbol{P}^{\text{NBV}}_{\boldsymbol{b}}; \sigma\right)}{\nabla \boldsymbol{P}^{\text{NBV}}_{\boldsymbol{b}}}$, to denoise the bounding box, rather than the point cloud, we take a further step following the chain rule ($\boldsymbol{P}^{\text{NBV}}_{\boldsymbol{b}}$ is differentiable according to Equation 2 of the main text):

$$\frac{\nabla \log p\left(\boldsymbol{b}; \sigma, \boldsymbol{P}\right)}{\nabla \boldsymbol{b}} = \frac{\nabla \log p\left(\boldsymbol{P}^{\text{NBV}}_{\boldsymbol{b}}; \sigma\right)}{\nabla \boldsymbol{P}^{\text{NBV}}_{\boldsymbol{b}}} \frac{\nabla \boldsymbol{P}^{\text{NBV}}_{\boldsymbol{b}}}{\nabla \boldsymbol{b}} \approx F_\theta\left(\boldsymbol{P}^{\text{NBV}}_{\boldsymbol{b}}; c_{\text{noise}}(\sigma)\right) \frac{\nabla \boldsymbol{P}^{\text{NBV}}_{\boldsymbol{b}}}{\nabla \boldsymbol{b}}. \tag{6}$$

Let $\boldsymbol{b}_T \sim p\left(\boldsymbol{b}; \sigma_{\text{max}}\right)$ be the imperfect bounding box predicted by an adapted object detector, and let $\boldsymbol{b}_0 \sim p\left(\boldsymbol{b}; 0\right)$ denote the corresponding box after refinement. Similarly, bounding box refinement can be achieved by evloving $\boldsymbol{b}_T$ to $\boldsymbol{b}_0$ following:

$$\mathrm{d}\boldsymbol{b} = -\dot{\sigma}(t)\sigma(t)\frac{\nabla \log p\left(\boldsymbol{b}; \sigma, \boldsymbol{P}\right)}{\nabla \boldsymbol{b}}\mathrm{d}t, \tag{7}$$

3.3.3 Shape Guidance

The probabilistic flow ODE allows adding objectives other than the score function to bounding box refinement without any retraining. For instance, [48] assumes the average object size $(\bar{w}, \bar{h}, \bar{l})$ in the target domain is available. Such information can be used to further improve domain adaptation performance by simply rewriting Equation 7 as:

$$\mathrm{d}\boldsymbol{b} = -\dot{\sigma}(t)\sigma(t)\left[\frac{\log p\left(\boldsymbol{b}; \sigma, \boldsymbol{P}\right)}{\nabla \boldsymbol{b}} + \alpha \frac{\nabla \ell_{\text{size}}\left(\boldsymbol{b}, \bar{w}, \bar{h}, \bar{l}\right)}{\nabla \boldsymbol{b}}\right] \mathrm{d}t, \tag{8}$$

where α denotes shape weight, and

$$\ell_{\text{size}}\left(\boldsymbol{b}, \bar{w}, \bar{h}, \bar{l}\right) = \|w - \bar{w}\|^2 + \|h - \bar{h}\|^2 + \|l - \bar{l}\|^2. \tag{9}$$

4 Experiments

4.1 Experimental Setup

Datasets. We primarily consider three datasets: The KITTI dataset [6], the Lyft Level 5 Perception dataset [16], and the Ithaca365 dataset [5]. For KITTI, we follow the official splits. For Lyft, we follow various existing works [58, 59, 24] and use the splits separated by geographical locations, consisting of 11,873 point clouds for training and 4,901 for testing. For Ithaca365, we utilize the annotated point clouds with 4,445 for training and 1,644 for testing. Additionally, we include experiments with the nuScenes dataset in the supplementary to evaluate DiffuBox's performance on larger-scale and more diverse data.

https://doi.org/10.52202/079017-3293

Table 1: **mAP@IoU 0.7 for KITTI → Lyft (cars).** Higher is Better. DiffuBox leads to improvement in almost all cases, with especially significant gain for the Direct and OT detections.

Method	BEV↑				3D↑			
	0-30m	30-50m	50-80m	0-80m	0-30m	30-50m	50-80m	0-80m
Direct	68.03	38.62	9.99	39.06	25.76	7.84	1.04	12.07
Direct+DiffuBox	**88.95**	**73.27**	**23.84**	**59.70**	**62.94**	**35.44**	**6.67**	**35.56**
OT	75.07	61.84	20.44	51.95	18.67	10.57	1.82	11.89
OT+DiffuBox	**92.67**	**74.46**	**23.95**	**60.98**	**50.99**	**33.06**	**6.87**	**31.21**
SN	92.88	69.97	**25.68**	61.67	**70.40**	32.96	6.18	36.64
SN+DiffuBox	**94.77**	**72.09**	25.47	**62.70**	69.62	**40.39**	**7.17**	**38.72**
Rote-DA	89.64	70.10	**27.96**	60.63	50.65	24.92	**7.43**	28.63
Rote-DA+DiffuBox	**95.10**	**75.10**	23.35	**62.14**	**71.00**	**48.89**	6.48	**41.06**
ST3D	72.86	64.23	34.96	55.58	35.22	26.33	6.06	22.47
ST3D+DiffuBox	**92.08**	**75.03**	**35.35**	**66.08**	**61.58**	**45.44**	**10.16**	**39.81**

Baselines. We consider five domain adaptation baselines: (1) directly applying an out-of-domain detector without adaptation (Direct); (2) Output Transformation (OT) [48]; (3) Statistical Normalization (SN) [48]; (4) Rote-DA [59]; (5) ST3D [53]. OT and SN perform resizing based on the average sizes from the target domain. OT directly resizes the predicted bounding boxes on the target domain, while SN trains the detector with resized objects and boxes from the source domain. Rote-DA and ST3D perform self-training. Rote-DA leverages an additional context in the form of persistency-prior [1] and enforces consistency across domains. ST3D leverages better data augmentation and a memory bank for high-quality detections. As DiffuBox is complementary to these methods, we compare the detection performance of these methods before and after refining with DiffuBox.

Evaluation Metrics. We evaluate the detection performance in Bird's Eye View (BEV) and 3D. At depth ranges of 0-30m, 30-50m and 50-80m, we report the mean Average Precision (mAP) with Intersection over Union (IoU) thresholds set at 0.7 for cars, and 0.5 for pedestrians and cyclists. We also consider the nuScenes true positive metrics [2]: translation error, scale error and rotation error. These measure the error in center offset, size difference, and orientation offset, respectively, of all true positive detections.

Implementation Details. We use the implementation and configurations from OpenPCDet [44] for detectors, and [14]'s implementation for diffusion models. We set the context limit to 4x the bounding box size. We use shape weight 0.1 for cars and pedestrians, and 0.01 for cyclists as cyclists have more shape variation. More details can be found in the supplementary.

4.2 Experimental Results

We present the results for KITTI → Lyft cars in Table 1 (mAP@IoU 0.7) and Table S8 (nuScenes TP metrics), and the results for KITTI → Ithaca365 cars in Table 2 (mAP@IoU 0.7) and Table S9 (nuScenes TP metrics). We additionally include KITTI → nuScenes results in Table S10 in the supplementary. We use PointRCNN [39] detectors; evaluations with other detectors can be found in the section below. DiffuBox consistently attains significant performance gain across different domain adaptation methods and datasets. The improvement of DiffuBox is especially significant for near-range and middle-range detections. Notably, for KITTI → Lyft cars, DiffuBox applied upon the *Direct* outputs is able to attain comparable performance with domain adaptation methods that require training such as ST3D. We hypothesize that this is because there are more LiDAR points for near-range and middle-range objects, which allows DiffuBox to better correct the detections.

DiffuBox with Other Detectors. To further demonstrate the robustness and versatility of DiffuBox, we present DiffuBox's performance for refining predictions from other detectors. We consider PointPillar [20], SECOND [51], PV-RCNN [38], CenterPoint [56] and DSVT [47] trained on KITTI. We report the mAP@IoU 0.7 for KITTI → Lyft cars in Table 3. Results show that DiffuBox consistently improves the predictions from different detectors.

Table 2: **mAP@IoU 0.7 for KITTI → Ithaca365 (cars).** DiffuBox leads to significant improvement upon different adaptation methods.

Method	BEV↑				3D↑			
	0-30m	30-50m	50-80m	0-80m	0-30m	30-50m	50-80m	0-80m
Direct	52.59	21.19	3.20	25.08	25.09	6.25	0.17	10.53
Direct+DiffuBox	**61.89**	**32.09**	**6.05**	**32.27**	**42.23**	**17.79**	**1.47**	**20.51**
OT	59.34	29.18	5.26	30.11	32.05	12.00	1.16	14.71
OT+DiffuBox	**60.76**	**32.56**	**6.07**	**31.89**	**40.43**	**18.33**	**1.61**	**19.80**
SN	60.48	31.04	**4.04**	29.80	32.17	13.03	0.85	15.02
SN+DiffuBox	**60.79**	**34.49**	3.79	**30.81**	**37.21**	**18.90**	**1.33**	**18.31**
Rote-DA	71.14	44.76	14.00	**42.38**	43.07	22.42	2.46	22.38
Rote-DA+DiffuBox	**71.52**	**45.44**	**14.56**	42.28	**46.77**	**25.72**	**4.17**	**25.02**

Table 3: **mAP@IoU 0.7 for KITTI → Lyft (cars) with other detectors.** DiffuBox consistently improves the detections from different detectors.

Method	BEV↑				3D↑			
	0-30m	30-50m	50-80m	0-80m	0-30m	30-50m	50-80m	0-80m
PointPillar (Direct)	65.77	39.02	11.28	36.80	16.58	5.06	0.56	6.87
PointPillar (Direct)+DiffuBox	**84.76**	**65.57**	**17.46**	**53.67**	**65.41**	**32.77**	**4.55**	**33.82**
SECOND (Direct)	65.61	38.83	13.92	38.06	24.39	8.32	0.86	10.68
SECOND (Direct)+DiffuBox	**89.70**	**66.98**	**19.11**	**57.04**	**61.90**	**32.24**	**4.39**	**33.19**
PV-RCNN (Direct)	73.56	46.39	13.63	43.72	34.20	14.03	1.53	16.17
PV-RCNN (Direct)+DiffuBox	**92.40**	**68.35**	**20.68**	**59.30**	**63.82**	**35.78**	**5.03**	**35.43**
PV-RCNN (OT)	80.48	54.40	17.54	51.03	19.87	8.15	0.75	10.74
PV-RCNN (OT)+DiffuBox	**93.65**	**69.06**	**21.35**	**60.16**	**53.68**	**31.31**	**4.00**	**30.63**
PV-RCNN (SN)	**94.16**	68.58	**22.22**	**62.16**	**72.72**	27.86	3.43	33.47
PV-RCNN (SN)+DiffuBox	93.99	**69.24**	21.13	61.23	67.83	**35.08**	**4.50**	**34.93**
CenterPoint (Direct)	74.91	36.64	2.47	36.23	28.63	4.05	0.15	10.29
CenterPoint (Direct)+DiffuBox	**90.25**	**58.65**	**6.95**	**51.38**	**71.02**	**34.91**	**1.48**	**34.40**
CenterPoint (OT)	82.81	51.02	6.26	46.10	25.34	12.35	0.52	13.34
CenterPoint (OT)+DiffuBox	**91.79**	**59.24**	**7.45**	**51.88**	**63.01**	**32.53**	**1.24**	**30.95**
DSVT (Direct)	68.93	47.49	11.32	41.77	33.72	11.74	1.42	15.67
DSVT (Direct)+DiffuBox	**89.01**	**63.41**	**17.50**	**55.27**	**65.22**	**36.31**	**5.02**	**35.61**
DSVT (OT)	71.85	42.93	13.18	43.05	15.66	4.47	0.31	8.06
DSVT (OT)+DiffuBox	**90.21**	**63.50**	**17.91**	**55.61**	**56.84**	**31.19**	**4.63**	**31.12**

DiffuBox on Other Object Classes. We present DiffuBox's performance on other object classes, specifically pedestrians and cyclists. We use the same configurations as cars, except cyclist shape weight 0.01 as mentioned in the implementation details. We report the KITTI → Lyft performance in Table 4, and the KITTI → Ithaca365 performance in Table S11. For Ithaca365, we only evaluate for pedestrians as Ithaca365 has very few cyclists. Results show that DiffuBox consistently attains significant improvement across object classes and domains, and can improve upon other domain adaptation methods. This shows the robustness and versatility of DiffuBox.

4.3 Qualitative Results

Figure 3 visualizes four scenes from the Lyft and Ithaca365 datasets. We compare the ground truth bounding boxes (green), the detections directly obtained from a PointRCNN trained on KITTI (yellow), and the refined detections using DiffuBox (blue). The out-of-domain PointRCNN produces reasonable results, but occasionally it produces false positives or boxes with incorrect shapes or alignment. DiffuBox effectively moves the incorrect boxes towards having better location, shape and alignment. Also observe that for the already accurate boxes, DiffuBox makes little change to them.

4.4 DiffuBox Extension: Detector Retraining

https://doi.org/10.52202/079017-3293

Table 4: **mAP@IoU 0.5 for KITTI → Lyft (pedestrians/cyclists).** DiffuBox attains consistent improvement across object classes and is able to improve upon other domain adaptation methods.

Class	Method	BEV↑				3D↑			
		0-30m	30-50m	50-80m	0-80m	0-30m	30-50m	50-80m	0-80m
Pedestrian	Direct	31.89	25.75	0.51	20.74	21.29	16.59	0.18	14.17
	Direct+DiffuBox	**43.64**	**26.97**	**0.61**	**25.10**	**34.30**	**22.54**	**0.33**	**20.46**
	OT	35.63	**25.47**	**0.64**	21.69	27.76	19.02	0.35	16.77
	OT+DiffuBox	**42.63**	25.28	0.55	**24.02**	**37.34**	**22.16**	**0.42**	**20.94**
	SN	43.75	**36.87**	0.67	28.18	34.12	26.08	0.45	21.00
	SN+DiffuBox	**50.98**	35.88	**1.08**	**29.45**	**38.55**	**30.24**	**0.60**	**23.50**
	Rote-DA	49.14	46.86	**1.23**	33.60	37.68	39.25	1.01	26.75
	Rote-DA+DiffuBox	**54.14**	**50.30**	1.22	**36.04**	**42.76**	**43.25**	**1.04**	**29.93**
Cyclist	Direct	48.50	8.91	**0.13**	26.96	38.13	5.19	**0.02**	21.11
	Direct+DiffuBox	**61.28**	**10.60**	0.06	**34.90**	**49.76**	**6.87**	**0.02**	**27.35**
	OT	55.37	9.99	**0.14**	30.90	20.09	4.11	**0.02**	10.96
	OT+DiffuBox	**65.93**	**10.26**	0.06	**36.86**	**32.93**	**6.11**	**0.02**	**18.81**
	SN	46.75	11.42	0.05	26.30	36.87	6.23	**0.03**	20.18
	SN+DiffuBox	**59.48**	**15.82**	**0.10**	**34.62**	**48.92**	**10.31**	0.02	**27.38**
	Rote-DA	77.19	**34.61**	**0.09**	48.66	70.35	**30.77**	**0.05**	44.29
	Rote-DA+DiffuBox	**83.80**	31.36	0.05	**51.49**	**75.79**	27.98	0.04	**45.77**

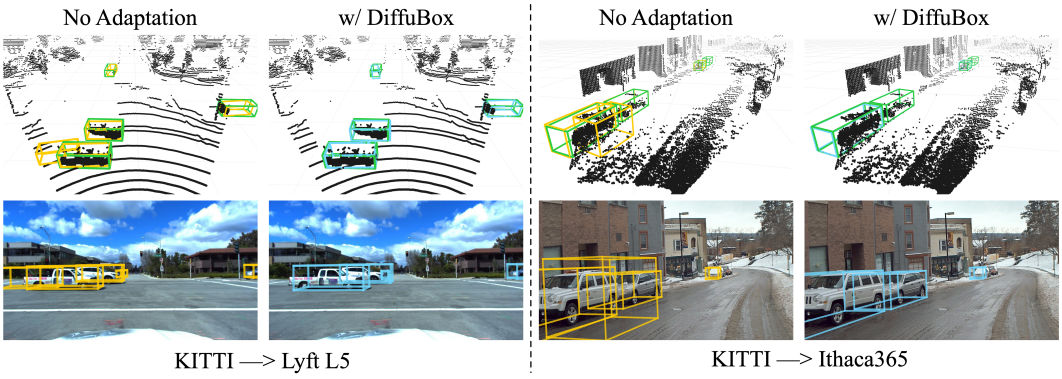


Figure 3: **Illustration of 3D object detection on Lyft/Ithaca365 before and after DiffuBox's refinement.** We visualize detections from an out-of-domain PointRCNN on four scenes from each dataset. We color the ground truth boxes in green, the detector outputs in yellow, and DiffuBox's refinements in blue. The out-of-domain detector sometimes produces false positives or boxes with incorrect shape or alignment. DiffuBox effectively improves the wrong or inaccurate boxes, while making little change to the accurate boxes.

One extension of DiffuBox for domain adaptation is to retrain detectors [57, 58] with DiffuBox's refined boxes. We can take a detector trained on the source domain, obtain its predictions on the target domain, refine the predictions with DiffuBox, and then retrain a detector using the refined boxes as labels.

We provide the KITTI → Lyft results in Table 5. We compare the performance of (1) directly applying the KITTI detector, (2) retraining for one round with the KITTI detector's predictions as labels, and (3) retraining for one round with the KITTI detector's predictions after DiffuBox's refinement as labels. Retraining is performed based on the KITTI detector's predictions on the Lyft training split, and the evaluation is conducted over Lyft's testing split. Retraining using

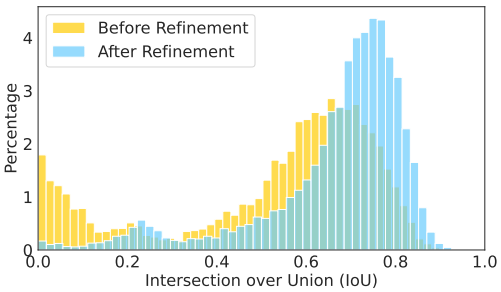


Figure 4: **Comparison of bounding box quality before and after refinement with DiffuBox.** We report the distribution of Intersection over Union (IoU) with ground-truth labels from the Lyft dataset. The unrefined predictions are from an unadapted Point-RCNN model trained on KITTI. We show that DiffuBox leads to significant improvement in bounding box localization.

the KITTI detector's predictions directly only provides limited improvement over directly applying the KITTI detector, while retraining with DiffuBox's refinement leads to significant improvement.

Table 5: **Retraining Performance (mAP@IoU 0.7) for KITTI → Lyft.** Retraining with DiffuBox's refined detections attains significant performance improvement.

Class	Method	BEV↑				3D↑			
		0-30m	30-50m	50-80m	0-80m	0-30m	30-50m	50-80m	0-80m
Car	Direct	68.03	38.62	9.99	39.06	25.76	7.84	1.04	12.07
	Retraining w/ Direct	70.98	45.53	15.12	43.49	30.45	10.54	1.69	14.55
	Retraining w/ Direct+DiffuBox	**91.79**	**77.52**	**35.33**	**64.71**	**67.80**	**36.17**	**10.56**	**37.55**
Pedestrian	Direct	31.89	25.75	0.51	20.74	21.29	16.59	0.18	14.17
	Retraining w/ Direct	34.23	34.65	**0.95**	24.37	27.58	30.33	**0.37**	20.22
	Retraining w/ Direct+DiffuBox	**44.17**	**39.39**	0.84	**29.22**	**35.60**	**34.67**	0.36	**24.71**
Cyclist	Direct	48.50	**8.91**	0.13	26.96	38.13	5.19	0.02	21.11
	Retraining w/ Direct	44.06	6.05	0.21	24.29	33.55	**5.73**	**0.14**	18.73
	Retraining w/ Direct+DiffuBox	**57.57**	6.54	**0.23**	**31.36**	**45.79**	5.19	0.11	**24.39**

4.5 Ablation Studies and Analysis

Context Limit. We conduct ablation study on the context limit, and consider ranges of 2x, 4x and 6x the box size. We evaluate under the setting of no adaptation (*Direct*) for KITTI → Lyft cars, and compare the mAP@IoU 0.7 in Table 6. All three context limits lead to significant performance improvement. Larger limit attains more gain, and the gain saturates at around the 4x limit.

Denoising Steps. We perform ablation on the number of denoising steps used, and present the results in Figure 5. In general, a majority of the performance is already reached using 8 diffusion steps, and it saturates around using 14 steps.

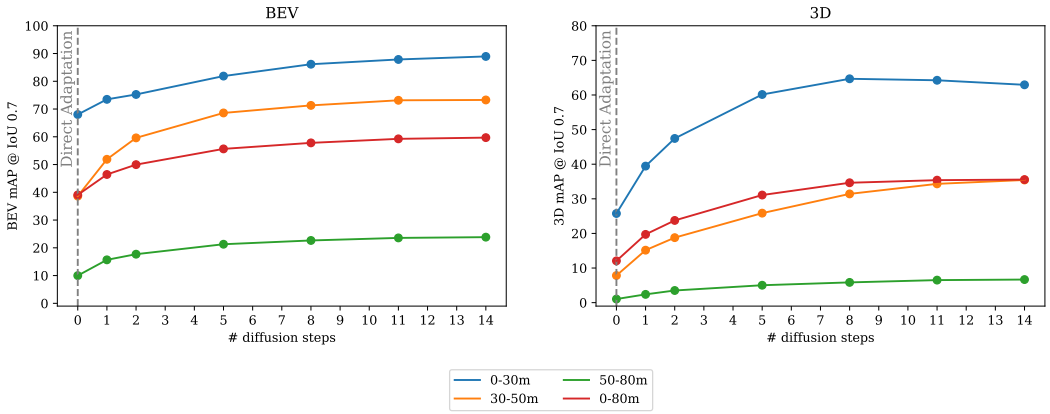


Figure 5: **mAP vs. Number of Diffusion Steps.** We report the BEV (left) and 3D (right) mAP @ IoU 0.7 for the setting of KITTI → Lyft Cars and PointRCNN detector.

Shape Weight. We perform ablation on shape guidance, and consider shape weight 0, 0.01, 0.1 and 0.5. We evaluate under the setting of no adaptation (*Direct*) for KITTI → Lyft cars and report the mAP@IoU 0.7 in Table 7. DiffuBox improves the detector's output even without shape weight. Using shape weights leads to additional gain, which saturates at around shape weight 0.1.

IoU Performance Analysis. We visualize the comparison of IoU of the bounding boxes with the ground truth bounding boxes both before and after using DiffuBox in Figure 4. The IoU of predictions after DiffuBox refinement (in blue) improves significantly over those before refinement (in yellow). This suggests that a majority of our refinement is in improving the bounding boxes' shape and alignment to fit into the new domain, thus resulting in higher IoU values.

https://doi.org/10.52202/079017-3293

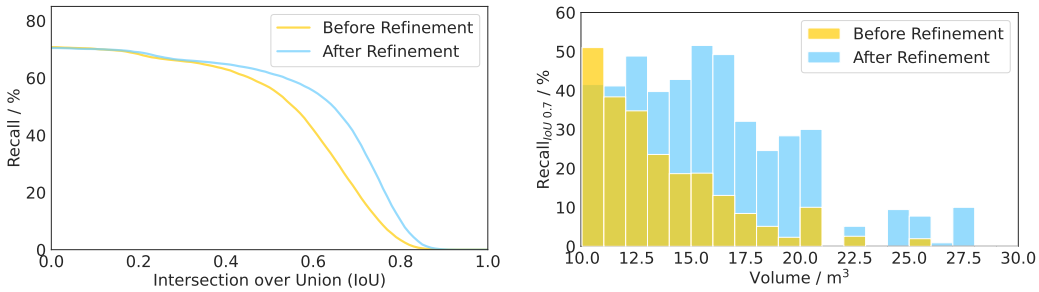
Table 6: **Ablation on Context Limit.** DiffuBox is robust against the choice of context limit. Larger limit could lead to better performance, with the gain saturated at around 4x limit.

Method	BEV↑				3D↑			
	0-30m	30-50m	50-80m	0-80m	0-30m	30-50m	50-80m	0-80m
Direct	68.03	38.62	9.99	39.06	25.76	7.84	1.04	12.07
Direct+DiffuBox 2x	83.16	68.91	17.65	55.53	54.63	27.63	3.62	29.51
Direct+DiffuBox 4x	88.95	**73.27**	23.84	59.70	**62.94**	**35.44**	**6.67**	**35.56**
Direct+DiffuBox 6x	**91.68**	72.56	**24.60**	**60.59**	61.92	30.91	5.39	33.61

Table 7: **Ablation on Shape Weight.** DiffuBox improves the detector's output significantly even without using shape weight, and using shape weight attains additional gain.

Method	BEV↑				3D↑			
	0-30m	30-50m	50-80m	0-80m	0-30m	30-50m	50-80m	0-80m
Direct	68.03	38.62	9.99	39.06	25.76	7.84	1.04	12.07
No SW	72.62	50.70	14.32	45.27	34.63	13.47	2.17	17.54
SW 0.01	74.81	57.48	17.18	48.89	47.11	17.74	2.99	23.10
SW 0.1	88.95	**73.27**	23.84	59.70	**62.94**	**35.44**	6.67	**35.56**
SW 0.5	**91.36**	73.09	**24.48**	**60.23**	50.70	32.73	**7.21**	30.54

Recall Analysis. We perform analysis on DiffuBox's effect on detection recall in Figure 6. As DiffuBox improves IoU for mislocalized detections, it reduces false negatives that arise from match IoU being lower than the threshold. The improvement is observed for objects across different sizes.



(a) Recall vs. IoU

(b) Recall vs. Object volume

Figure 6: **Recall improvement with DiffuBox.** We report recall on the *car* class (KITTI → Lyft, PointRCNN) before and after refinement with DiffuBox.

5 Discussion and Future Work

In this work, we propose DiffuBox, a diffusion-based approach that refines bounding boxes for better domain adaptation. While DiffuBox effectively improves the existing bounding boxes, one limitation is that DiffuBox currently does not consider false negatives that are due to completely missed detections. This could potentially be addressed through further distilling the detectors by DiffuBox's refined boxes, or by incorporating exploration strategies to capture possibly missed objects. Alternatively, a view that we did not discuss in this work, but is of potential interest to the field, is the use of DiffuBox for automatic label refinement. This can be useful for correcting slightly mis-aligned boxes, or labels across sensors that may have slightly unsynchronized sensors. We leave this discussion for further work, and will provide code and model checkpoints for this use case. With our work, we do not foresee any negative societal impacts and hope the field continues to develop such label refinement methods for 3D object detection.

Acknowledgments and Disclosure of Funding

This research is supported by grants from the National Science Foundation NSF (IIS-2107161, and IIS-1724282, HDR-2118310). The Cornell Center for Materials Research with funding from the NSF MRSEC program (DMR-1719875), DARPA, arXiv, LinkedIn, Nvidia, and the New York Presbyterian Hospital.

References

[1] Dan Barnes, Will Maddern, Geoffrey Pascoe, and Ingmar Posner. Driven to distraction: Self-supervised distractor learning for robust monocular visual odometry in urban environments. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1894–1900. IEEE, 2018. 6

[2] 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*, 2020. 6

[3] Shoufa Chen, Peize Sun, Yibing Song, and Ping Luo. Diffusiondet: Diffusion model for object detection. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 19830–19843, 2023. 3

[4] Prafulla Dhariwal and Alexander Nichol. Diffusion models beat gans on image synthesis. *Advances in neural information processing systems*, 34:8780–8794, 2021. 3

[5] Carlos A Diaz-Ruiz, Youya Xia, Yurong You, Jose Nino, Junan Chen, Josephine Monica, Xiangyu Chen, Katie Luo, Yan Wang, Marc Emond, et al. Ithaca365: Dataset and driving perception under repeated and challenging weather conditions. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 21383–21392, 2022. 2, 5

[6] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. Vision meets robotics: The kitti dataset. *The International Journal of Robotics Research*, 32(11):1231–1237, 2013. 5

[7] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012. 2, 3, 19

[8] Jonathan Ho, William Chan, Chitwan Saharia, Jay Whang, Ruiqi Gao, Alexey Gritsenko, Diederik P. Kingma, Ben Poole, Mohammad Norouzi, David J. Fleet, and Tim Salimans. Imagen video: High definition video generation with diffusion models, 2022. 3

[9] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020. 3

[10] Jonathan Ho, Tim Salimans, Alexey Gritsenko, William Chan, Mohammad Norouzi, and David J. Fleet. Video diffusion models, 2022. 3

[11] Edward L Ince. *Ordinary differential equations*. Courier Corporation, 1956. 4

[12] Peng Jiang and Srikanth Saripalli. Lidarnet: A boundary-aware domain adaptation model for point cloud semantic segmentation, 2021. 3

[13] Tero Karras, Miika Aittala, Timo Aila, and Samuli Laine. Elucidating the design space of diffusion-based generative models. *Advances in Neural Information Processing Systems*, 35:26565–26577, 2022. 4, 15

[14] Tero Karras, Miika Aittala, Timo Aila, and Samuli Laine. Elucidating the design space of diffusion-based generative models. *Advances in Neural Information Processing Systems*, 35:26565–26577, 2022. 6, 15

[15] R. Kesten, M. Usman, J. Houston, T. Pandya, K. Nadhamuni, A. Ferreira, M. Yuan, B. Low, A. Jain, P. Ondruska, S. Omari, S. Shah, A. Kulkarni, A. Kazakova, C. Tao, L. Platinsky, W. Jiang, and V. Shet. Lyft level 5 av dataset 2019. urlhttps://level5.lyft.com/dataset/, 2019. 2, 3, 19

[16] R. Kesten, M. Usman, J. Houston, T. Pandya, K. Nadhamuni, A. Ferreira, M. Yuan, B. Low, A. Jain, P. Ondruska, S. Omari, S. Shah, A. Kulkarni, A. Kazakova, C. Tao, L. Platinsky, W. Jiang, and V. Shet. Lyft level 5 av dataset 2019. urlhttps://level5.lyft.com/dataset/, 2019. 5

[17] Hyeonseong Kim, Yoonsu Kang, Changgyoon Oh, and Kuk-Jin Yoon. Single domain generalization for lidar semantic segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 17587–17598, June 2023. 3

 https://doi.org/10.52202/079017-3293

[18] Se-Ho Kim, Inyong Koo, Inyoung Lee, Byeongjun Park, and Changick Kim. Diffref3d: A diffusion-based proposal refinement framework for 3d object detection. *arXiv preprint arXiv:2310.16349*, 2023. 3

[19] Lingdong Kong, Niamul Quader, and Venice Erin Liong. Conda: Unsupervised domain adaptation for lidar segmentation via regularized domain concatenation, 2023. 3

[20] Alex H Lang, Sourabh Vora, Holger Caesar, Lubing Zhou, Jiong Yang, and Oscar Beijbom. Pointpillars: Fast encoders for object detection from point clouds. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 12697–12705, 2019. 2, 6

[21] Ferdinand Langer, Andres Milioto, Alexandre Haag, Jens Behley, and C. Stachniss. Domain transfer for semantic segmentation of lidar data using deep neural networks. *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 8263–8270, 2020. 3

[22] Jinlong Li, Runsheng Xu, Jin Ma, Qin Zou, Jiaqi Ma, and Hongkai Yu. Domain adaptive object detection for autonomous driving under foggy weather, 2022. 3

[23] Katie Luo, Zhenzhen Liu, Xiangyu Chen, Yurong You, Sagie Benaim, Cheng Perng Phoo, Mark Campbell, Wen Sun, Bharath Hariharan, and Kilian Q Weinberger. Reward finetuning for faster and more accurate unsupervised object discovery. *Advances in Neural Information Processing Systems*, 36, 2024. 1, 3

[24] Katie Z Luo, Zhenzhen Liu, Xiangyu Chen, Yurong You, Sagie Benaim, Cheng Perng Phoo, Mark Campbell, Wen Sun, Bharath Hariharan, and Kilian Q Weinberger. Reward finetuning for faster and more accurate unsupervised object discovery. *arXiv preprint arXiv:2310.19080*, 2023. 5

[25] 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*, pages 2837–2845, 2021. 3, 4

[26] Jiageng Mao, Yujing Xue, Minzhe Niu, Haoyue Bai, Jiashi Feng, Xiaodan Liang, Hang Xu, and Chunjing Xu. Voxel transformer for 3d object detection. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3164–3173, 2021. 2

[27] Luke Melas-Kyriazi, Christian Rupprecht, and Andrea Vedaldi. Pc2: Projection-conditioned point cloud diffusion for single-image 3d reconstruction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 12923–12932, June 2023. 3

[28] Pietro Morerio, Jacopo Cavazza, and Vittorio Murino. Minimal-entropy correlation alignment for unsupervised deep domain adaptation, 2017. 3

[29] Xuran Pan, Zhuofan Xia, Shiji Song, Li Erran Li, and Gao Huang. 3d object detection with pointformer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7463–7472, 2021. 2

[30] Ben Poole, Ajay Jain, Jonathan T Barron, and Ben Mildenhall. Dreamfusion: Text-to-3d using 2d diffusion. *arXiv preprint arXiv:2209.14988*, 2022. 4

[31] Charles R Qi, Wei Liu, Chenxia Wu, Hao Su, and Leonidas J Guibas. Frustum pointnets for 3d object detection from rgb-d data. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 918–927, 2018. 2

[32] 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*, pages 652–660, 2017. 2

[33] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. *Advances in neural information processing systems*, 30, 2017. 2

[34] 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*, pages 10684–10695, 2022. 3

[35] Kuniaki Saito, Yoshitaka Ushiku, Tatsuya Harada, and Kate Saenko. Adversarial dropout regularization, 2018. 3

[36] Khaled Saleh, Ahmed Abobakr, Mohammed Attia, Julie Iskander, Darius Nahavandi, and Mohammed Hossny. Domain adaptation for vehicle detection from bird's eye view lidar point cloud data, 2019. 3

[37] Cristiano Saltori, Fabio Galasso, Giuseppe Fiameni, Nicu Sebe, Elisa Ricci, and Fabio Poiesi. Cosmix: Compositional semantic mix for domain adaptation in 3d lidar segmentation, 2022. 3

[38] Shaoshuai Shi, Chaoxu Guo, Li Jiang, Zhe Wang, Jianping Shi, Xiaogang Wang, and Hongsheng Li. Pv-rcnn: Point-voxel feature set abstraction for 3d object detection. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10529–10538, 2020. 2, 6

[39] Shaoshuai Shi, Xiaogang Wang, and Hongsheng Li. Pointrcnn: 3d object proposal generation and detection from point cloud. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 770–779, 2019. 2, 6, 19

[40] Weijing Shi and Raj Rajkumar. Point-gnn: Graph neural network for 3d object detection in a point cloud. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 1711–1719, 2020. 2

[41] Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *International conference on machine learning*, pages 2256–2265. PMLR, 2015. 3

[42] Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. *arXiv preprint arXiv:2010.02502*, 2020. 3

[43] Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. *arXiv preprint arXiv:2011.13456*, 2020. 3, 4

[44] OpenPCDet Development Team. Openpcdet: An open-source toolbox for 3d object detection clouds. https://github.com/open-mmlab/OpenPCDet, 2020. 6, 15

[45] Nicolaas G Van Kampen. Stochastic differential equations. *Physics reports*, 24(3):171–228, 1976. 4

[46] Tuan-Hung Vu, Himalaya Jain, Maxime Bucher, Matthieu Cord, and Patrick Pérez. Advent: Adversarial entropy minimization for domain adaptation in semantic segmentation, 2019. 3

[47] Haiyang Wang, Chen Shi, Shaoshuai Shi, Meng Lei, Sen Wang, Di He, Bernt Schiele, and Liwei Wang. Dsvt: Dynamic sparse voxel transformer with rotated sets. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13520–13529, 2023. 6, 16, 17

[48] Yan Wang, Xiangyu Chen, Yurong You, Li Erran Li, Bharath Hariharan, Mark Campbell, Kilian Q Weinberger, and Wei-Lun Chao. Train in germany, test in the usa: Making 3d object detectors generalize. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11713–11723, 2020. 1, 2, 3, 5, 6, 19

[49] Yue Wang, Alireza Fathi, Abhijit Kundu, David A Ross, Caroline Pantofaru, Tom Funkhouser, and Justin Solomon. Pillar-based object detection for autonomous driving. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XXII 16*, pages 18–34. Springer, 2020. 2

[50] Chenfeng Xu, Bichen Wu, Zining Wang, Wei Zhan, Peter Vajda, Kurt Keutzer, and Masayoshi Tomizuka. Squeezesegv3: Spatially-adaptive convolution for efficient point-cloud segmentation, 2021. 3

[51] Yan Yan, Yuxing Mao, and Bo Li. Second: Sparsely embedded convolutional detection. *Sensors*, 18(10):3337, 2018. 2, 6

[52] Guandao Yang, Xun Huang, Zekun Hao, Ming-Yu Liu, Serge Belongie, and Bharath Hariharan. Pointflow: 3d point cloud generation with continuous normalizing flows. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 4541–4550, 2019. 4

[53] Jihan Yang, Shaoshuai Shi, Zhe Wang, Hongsheng Li, and Xiaojuan Qi. St3d: Self-training for unsupervised domain adaptation on 3d object detection. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10368–10378, 2021. 2, 3, 6

[54] Jihan Yang, Shaoshuai Shi, Zhe Wang, Hongsheng Li, and Xiaojuan Qi. St3d++: Denoised self-training for unsupervised domain adaptation on 3d object detection. *IEEE transactions on pattern analysis and machine intelligence*, 45(5):6354–6371, 2022. 2

[55] Zetong Yang, Yanan Sun, Shu Liu, and Jiaya Jia. 3dssd: Point-based 3d single stage object detector. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11040–11048, 2020. 2

[56] Tianwei Yin, Xingyi Zhou, and Philipp Krahenbuhl. Center-based 3d object detection and tracking. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11784–11793, 2021. 6, 16, 17

 https://doi.org/10.52202/079017-3293

[57] Yurong You, Carlos Andres Diaz-Ruiz, Yan Wang, Wei-Lun Chao, Bharath Hariharan, Mark Campbell, and Kilian Q Weinbergert. Exploiting playbacks in unsupervised domain adaptation for 3d object detection in self-driving cars. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 5070–5077. IEEE, 2022. 8

[58] Yurong You, Katie Luo, Cheng Perng Phoo, Wei-Lun Chao, Wen Sun, Bharath Hariharan, Mark Campbell, and Kilian Q Weinberger. Learning to detect mobile objects from lidar scans without labels. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1130–1140, 2022. 5, 8

[59] Yurong You, Cheng Perng Phoo, Katie Luo, Travis Zhang, Wei-Lun Chao, Bharath Hariharan, Mark Campbell, and Kilian Q Weinberger. Unsupervised adaptation from repeated traversals for autonomous driving. *Advances in Neural Information Processing Systems*, 35:27716–27729, 2022. 2, 3, 5, 6

[60] Yurong You, Cheng Perng Phoo, Katie Luo, Travis Zhang, Wei-Lun Chao, Bharath Hariharan, Mark Campbell, and Kilian Q Weinberger. Unsupervised adaptation from repeated traversals for autonomous driving. *Advances in Neural Information Processing Systems*, 35:27716–27729, 2022. 19

[61] Xiaohui Zeng, Arash Vahdat, Francis Williams, Zan Gojcic, Or Litany, Sanja Fidler, and Karsten Kreis. Lion: Latent point diffusion models for 3d shape generation. *arXiv preprint arXiv:2210.06978*, 2022. 3, 4

[62] Linqi Zhou, Yilun Du, and Jiajun Wu. 3d shape generation and completion through point-voxel diffusion. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 5826–5835, October 2021. 3

[63] Linqi Zhou, Yilun Du, and Jiajun Wu. 3d shape generation and completion through point-voxel diffusion. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 5826–5835, 2021. 4

[64] Xin Zhou, Jinghua Hou, Tingting Yao, Dingkang Liang, Zhe Liu, Zhikang Zou, Xiaoqing Ye, Jianwei Cheng, and Xiang Bai. Diffusion-based 3d object detection with random boxes. *arXiv preprint arXiv:2309.02049*, 2023. 3

[65] Yin Zhou and Oncel Tuzel. Voxelnet: End-to-end learning for point cloud based 3d object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4490–4499, 2018. 2

Supplementary Material: DiffuBox: Refining 3D Object Detection with Point Diffusion

S1 Algorithmic Description of DiffuBox

Below we provide an algorithmic description of the training and inference workflow of DiffuBox.

Algorithm 1 DiffuBox Training

Require: Train set $\mathcal{D}_{\text{train}}$, noise distribution p_{train}
Ensure: Diffusion model F_θ to approximate $\nabla_{\boldsymbol{x}} \log p(\boldsymbol{x}, \sigma)$ of bounding box over points $\boldsymbol{x}$, noise σ

```
repeat
    Sample point cloud P, true bounding box b* where P, b* ~ D_train
    Sample noise level σ ~ p_train
    Compute P_{b*}^NBV from Eq. 2
    Update F_θ with diffusion objective Eq. 4
until convergence
```

Algorithm 2 DiffuBox Inference

Require: Set of initial bounding boxes $\{\boldsymbol{b}\}_{\text{scene}}$, point cloud $\boldsymbol{P}$, F_θ approximating $\nabla_{\boldsymbol{x}} \log p(\boldsymbol{x}, \sigma)$
Ensure: Set of refined bounding boxes $\{\hat{\boldsymbol{b}}\}_{\text{scene}}$

```
for each bounding box b in {b}_scene do
    for denoising steps t = 1, ··· , T do
        Obtain σ(t) from denoising schedule
        Compute P_b^NBV from Eq. 2
        Compute db with σ(t) and F_θ from Eq. 8
        b ← b + db
    end for
    b̂ ← b
end for
{b̂}_scene ← NMS({b̂}_scene)
```

S2 Implementation Details

Experimental Setup. For detectors, we use the implementation and configurations from OpenPCDet [44]. For diffusion models, we use [14]'s implementation and follow their noise schedule $\sigma_{max} = 80$. During inference, we denoise each box for 14 steps, starting from a noise level between $[10, 80]$ linear in the detector confidence; higher confidence boxes begin from smaller noise. For each bounding box, we consider the points that fall within the range of four times its size. Additionally, inspired by OT and SN, we slightly regularize the bounding box shape with the average size of the target domain, with shape weight 0.1 for cars and pedestrians, and 0.01 for cyclists. We use NVIDIA A6000 for all of our experiments.

Training Details. In the diffusion process, we follow [13] and use noise level distribution $\ln \sigma \sim \mathcal{N}\left(-1.2, 1.2^2\right)$, ODE schedule $\sigma(t) = t$, and 2nd order Heun solver. The denoiser transformer model contains 12 self-attention layers with hidden size 1024; each layer has 2048 intermediate dimensions and 8 heads. The diffusion model is trained with batch size 128 and learning rate 0.0001 for 100k steps. A visualization of our Diffusion model architecture can be found in Figure S7.

S3 Additional Experiment Results

S3.1 nuScenes True Positive Metrics

We report the nuScenes TP metrics for KITTI → Lyft cars in Table S8, and those for KITTI → Ithaca365 in Table S9. All results are based on PointRCNN detectors. DiffuBox effectively reduces all three types of errors, with especially significant improvement for translation and scale errors.

S3.2 Additional Experimental Results

We report the performance on KITTI → nuScenes cars in Table S10, and KITTI → Ithaca365 pedestrians in Table S11. Both sets of experiments are based on PointRCNN detectors. For nuScenes,

https://doi.org/10.52202/079017-3293

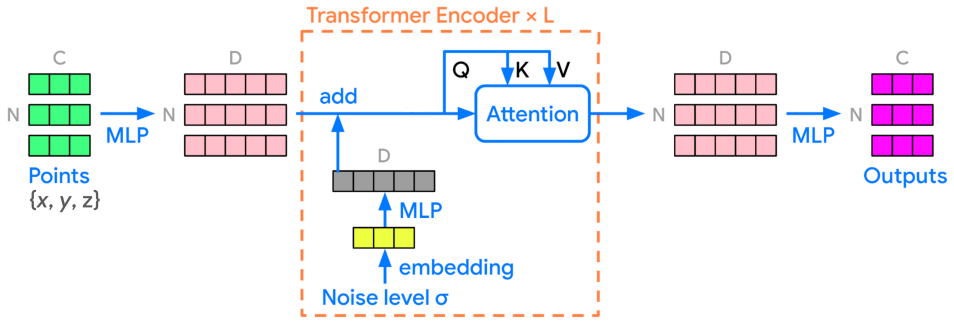


Figure S7: Architecture overview of DiffuBox's denoiser model. The model is composed of 2 MLP layers and L transformer encoder layers, which maps 3D points to a higher dimensional space for effective self-attention.

Table S8: **nuScenes TP metrics for KITTI → Lyft (cars).** Lower is better. DiffuBox reduces all three types of errors, with especially significant improvement for translation and scale errors.

Method	Translation Error↓				Scale Error↓				Orientation Error↓			
	0-30m	30-50m	50-80m	0-80m	0-30m	30-50m	50-80m	0-80m	0-30m	30-50m	50-80m	0-80m
Direct	0.346	0.473	0.506	0.430	0.267	0.299	0.309	0.289	0.347	0.510	0.624	0.472
Direct+DiffuBox	**0.221**	**0.289**	**0.424**	**0.293**	**0.162**	**0.171**	**0.198**	**0.173**	**0.329**	**0.473**	**0.607**	**0.447**
OT	0.273	0.376	0.466	0.356	0.217	0.207	0.219	0.214	0.338	0.496	0.623	0.463
OT+DiffuBox	**0.201**	**0.266**	**0.399**	**0.271**	**0.174**	**0.173**	**0.202**	**0.180**	**0.328**	**0.474**	**0.598**	**0.444**
SN	0.332	0.436	0.440	0.398	**0.160**	0.178	**0.204**	**0.178**	0.320	0.540	0.678	0.493
SN+DiffuBox	**0.201**	**0.251**	**0.410**	**0.274**	0.164	**0.171**	0.207	**0.178**	**0.309**	**0.505**	**0.631**	**0.463**
Rote-DA	0.286	0.352	**0.408**	0.338	0.203	0.213	**0.190**	0.204	0.253	0.448	0.621	0.408
Rote-DA+DiffuBox	**0.206**	**0.260**	0.423	**0.275**	**0.161**	**0.170**	0.201	**0.173**	**0.244**	**0.437**	**0.609**	**0.398**
ST3D	0.386	0.489	0.451	0.440	0.240	0.225	0.239	0.234	0.352	0.562	0.693	0.517
ST3D+DiffuBox	**0.216**	**0.262**	**0.350**	**0.267**	**0.162**	**0.171**	**0.199**	**0.175**	**0.326**	**0.508**	**0.681**	**0.485**

Table S9: **nuScenes TP metrics for KITTI → Ithaca365 (cars).** Lower is better.

Method	Translation Error↓				Scale Error↓				Orientation Error↓			
	0-30m	30-50m	50-80m	0-80m	0-30m	30-50m	50-80m	0-80m	0-30m	30-50m	50-80m	0-80m
Direct	0.403	0.523	**0.722**	0.510	0.200	0.206	0.207	0.204	**0.492**	**0.827**	**1.001**	**0.705**
Direct+DiffuBox	**0.312**	**0.456**	0.775	**0.458**	**0.125**	**0.131**	**0.172**	**0.137**	0.493	0.831	1.002	0.708
OT	0.381	0.506	**0.720**	0.494	0.159	0.150	**0.139**	0.152	**0.492**	**0.827**	1.004	0.707
OT+DiffuBox	**0.320**	**0.458**	0.753	**0.456**	**0.124**	**0.133**	0.173	**0.138**	0.494	0.829	**0.997**	**0.705**
SN	0.484	0.544	**0.816**	0.587	0.153	0.141	**0.150**	0.149	0.590	0.869	1.091	0.800
SN+DiffuBox	**0.351**	**0.492**	0.862	**0.520**	**0.128**	**0.137**	0.179	**0.144**	**0.574**	**0.862**	**1.072**	**0.783**
Rote-DA	0.492	0.484	**0.461**	0.482	0.140	0.144	0.167	0.147	0.476	0.731	1.023	0.683
Rote-DA+DiffuBox	**0.315**	**0.384**	0.517	**0.383**	**0.121**	**0.116**	**0.135**	**0.123**	**0.451**	**0.715**	**0.984**	**0.656**

we evaluate for up to 50m based on the point cloud range. DiffuBox attains significant improvement upon the detections in both settings.

Table S10: **mAP@IoU 0.7 for KITTI → nuScenes (cars).** DiffuBox consistently attains significant improvement.

Method	BEV ↑			3D ↑		
	0-30m	30-50m	0-50m	0-30m	30-50m	0-50m
Direct	44.78	0.70	15.86	14.82	0.00	4.66
Direct+DiffuBox	**58.07**	**1.06**	**20.70**	**22.77**	0.00	**7.40**

In addition, we report the full KITTI → Lyft performance on all three traffic participant classes with the CenterPoint [56] detector in Table S12 and DSVT [47] in Table S13. DiffuBox achieves consistent improvement across object detectors, classes, and domain adaptation methods.

Table S11: **mAP@IoU 0.5 for KITTI → Ithaca365 (pedestrians).** DiffuBox consistently attains significant improvement.

Method	BEV ↑				3D ↑			
	0-30m	30-50m	50-80m	0-80m	0-30m	30-50m	50-80m	0-80m
Direct	41.70	16.87	**1.63**	23.07	31.38	10.06	**0.60**	16.09
Direct+DiffuBox	**50.53**	**20.26**	0.49	**26.62**	**43.36**	**14.74**	0.10	**21.74**
OT	42.96	18.66	**1.75**	23.69	33.00	11.64	**0.63**	17.15
OT+DiffuBox	**50.52**	**21.27**	0.38	**26.95**	**44.67**	**15.45**	0.05	**22.48**
SN	48.95	16.97	**2.46**	26.36	38.47	9.17	**0.97**	17.89
SN+DiffuBox	**57.32**	**17.27**	0.51	**28.20**	**47.63**	**10.44**	0.22	**21.62**
Rote-DA	43.36	1.68	0.00	14.18	30.21	0.13	0.00	8.40
Rote-DA+DiffuBox	**56.91**	**21.31**	**1.18**	**29.75**	**50.94**	**14.63**	**0.31**	**24.18**

Table S12: **mAP for KITTI → Lyft with CenterPoint [56].** DiffuBox attains consistent improvement across object classes and domain adaptation methods.

Class	Method	BEV ↑				3D ↑			
		0-30m	30-50m	50-80m	0-80m	0-30m	30-50m	50-80m	0-80m
Car@IoU 0.7	Direct	74.91	36.64	2.47	36.23	28.63	4.05	0.15	10.29
	Direct+DiffuBox	**90.25**	**58.65**	**6.95**	**51.38**	**71.02**	**34.91**	**1.48**	**34.40**
	OT	82.81	51.02	6.26	46.10	25.34	12.35	0.52	13.34
	OT+DiffuBox	**91.79**	**59.24**	**7.45**	**51.88**	**63.01**	**32.53**	**1.24**	**30.95**
Pedestrian@IoU 0.5	Direct	0.85	1.22	0.02	0.49	0.41	0.47	0.01	0.19
	Direct+DiffuBox	**5.58**	**2.06**	**0.03**	**1.78**	**3.99**	**1.75**	**0.02**	**1.20**
	OT	1.88	2.02	**0.05**	0.87	0.61	1.22	0.01	0.37
	OT+DiffuBox	**5.78**	**2.32**	0.03	**1.78**	**4.04**	**1.90**	**0.02**	**1.16**
Cyclist@IoU 0.5	Direct	23.34	0.84	0.01	8.19	16.53	0.34	0.00	5.42
	Direct+DiffuBox	**42.22**	**1.49**	**0.02**	**15.22**	**31.74**	**0.58**	**0.01**	**11.28**
	OT	27.40	0.95	0.01	9.73	7.32	0.05	**0.01**	1.98
	OT+DiffuBox	**42.22**	**1.54**	**0.02**	**15.23**	**31.35**	**0.67**	**0.01**	**11.20**

Table S13: **mAP for KITTI → Lyft with DSVT [47].** DiffuBox attains consistent improvement across object classes and domain adaptation methods.

Class	Method	BEV ↑				3D ↑			
		0-30m	30-50m	50-80m	0-80m	0-30m	30-50m	50-80m	0-80m
Car@IoU 0.7	Direct	68.93	47.49	11.32	41.77	33.72	11.74	1.42	15.67
	Direct+DiffuBox	**89.01**	**63.41**	**17.50**	**55.27**	**65.22**	**36.31**	**5.02**	**35.61**
	OT	71.85	42.93	13.18	43.05	15.66	4.47	0.31	8.06
	OT+DiffuBox	**90.21**	**63.50**	**17.91**	**55.61**	**56.84**	**31.19**	**4.63**	**31.12**
Pedestrian@IoU 0.5	Direct	16.42	6.48	0.47	8.09	11.49	4.41	0.10	5.32
	Direct+DiffuBox	**27.89**	**8.13**	**1.05**	**12.42**	**23.55**	**6.49**	**0.53**	**10.28**
	OT	20.89	7.47	1.06	10.65	14.83	5.46	0.33	6.84
	OT+DiffuBox	**27.68**	**7.84**	**1.25**	**12.18**	**25.67**	**6.66**	**0.68**	**10.64**
Cyclist@IoU 0.5	Direct	38.46	2.04	0.00	19.82	30.16	1.56	0.00	15.65
	Direct+DiffuBox	**54.41**	**3.19**	0.00	**28.01**	**47.57**	**1.94**	0.00	**24.06**
	OT	43.43	2.18	0.00	22.30	14.65	1.23	0.00	8.01
	OT+DiffuBox	**55.26**	**3.04**	0.00	**28.75**	**38.12**	**1.89**	0.00	**19.52**

S3.3 Additional Qualitative Results

We present additional visualizations of box refinement over denoising steps, in both original and normalized views, in Figure S8. We show the box refinement process of the initial noisy prediction across different denoising steps, up to 14 steps. In addition, we show the points from the normalized box view (NBV) from the perspective of the refined boxes. Observe how the points of the car (colored in black) gradually get corrected to be inside the bounding box.

https://doi.org/10.52202/079017-3293

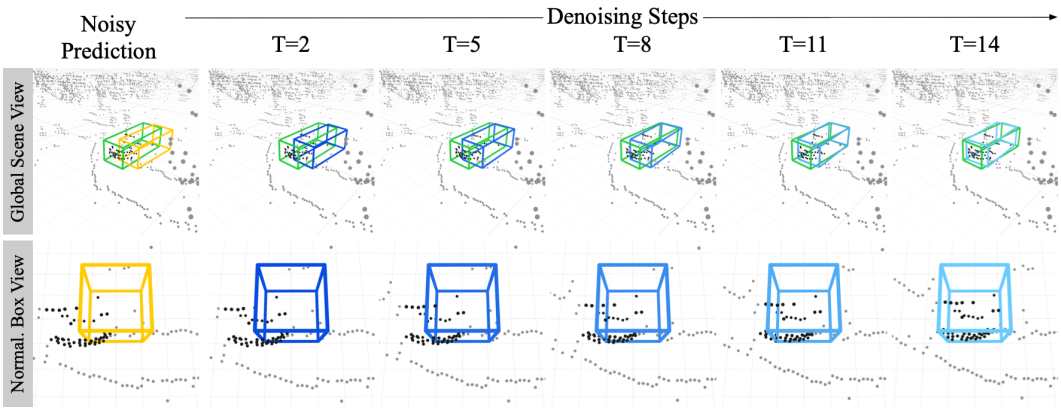


Figure S8: **Box refinement through denoising steps.** We visualize the correction of a noisy prediction, shown in yellow, using DiffuBox, as well as the normalized box view. The detection output is refined iteratively though the denoising steps, resulting in the final, corrected output of our method.

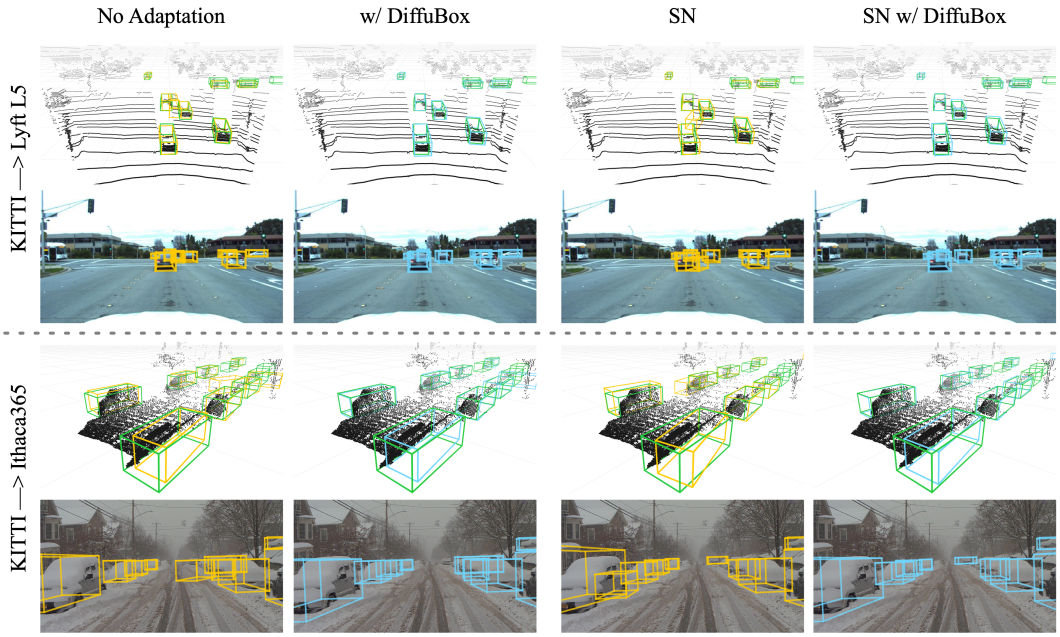


Figure S9: **Additional Qualitative Results of DiffuBox**. We include additional visualizations of 3D object detection on Lyft/Ithaca365 before and after DiffuBox's refinement. We color the ground truth boxes in green, the detector outputs in yellow, and the boxes refined with DiffuBox in blue.

We present additional visualizations of detections before and after using DiffuBox on another set of scene in Figure S9, and on different classes of traffic participants in Figure S10. Observe that DiffuBox's refinement is able to better align the boxes, and the corrected boxes can reduce false positives as they get corrected into the same set of points.

S3.4 Improvement Expectations and Upper Bounds

We validate our claims that improvements in localization is the major source of performance gain in Table S14. In particular, we show that large gains in performance can be obtained – across multiple domain adaptation algorithms – if localization is corrected to the ground truth position. This suggests that there is significant room for improvement in object detection performance from better localization, which DiffuBox aims to tackle.

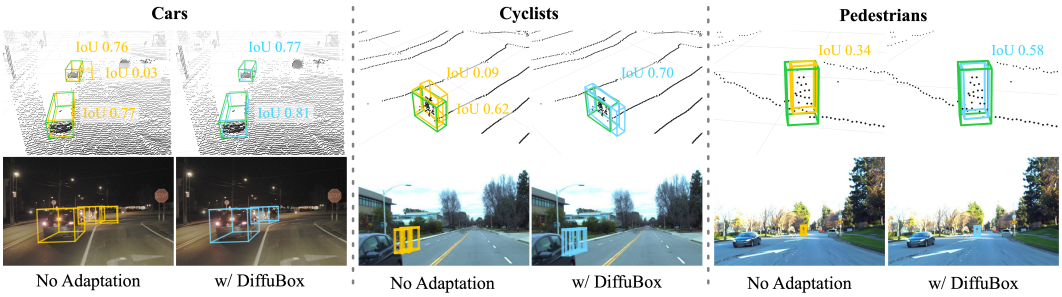


Figure S10: **Additional Qualitative Results of DiffuBox for other classes**. We include additional visualizations of 3D object detection for different classes. We color the ground truth boxes in green, the detector outputs in yellow, and the boxes refined with DiffuBox in blue.

Table S14: **Domain adaptation performance and potential improvement from correcting localization.** We report the performance of multiple domain adaptation algorithms (including *Direct*ly applying the model) on a Point-RCNN [39] detector adapted from KITTI [7] to Lyft [15] dataset. We evaluate mean Average Precision (mAP) on the *Car* category at $\text{IoU}_{3D} = 0.7$. *Oracle Loc* means assigning ground-truth bounding boxes to any intersecting detections (*i.e.*, infinitesimal IoU_{3D})

Method	0-30m	30-50m	50-80m	0-80m
Direct	25.76	7.84	1.04	12.07
Direct+Oracle Loc.	**95.50**	**80.63**	**38.24**	**68.24**
Rote-DA [60]	50.63	24.80	7.07	29.19
Rote-DA+Oracle Loc.	**98.54**	**80.15**	**38.55**	**70.80**
SN [48]	70.40	32.96	6.18	36.64
SN+Oracle Loc.	**96.90**	**78.19**	**38.75**	**70.36**

S3.5 Further Discussion

Interpretation of the effectiveness of the Statistical Normalization (SN) method. Recent domain adaptation algorithms typically rely on Statistical Normalization (SN) [48] to tackle the domain bias in object size. SN's solution is to adapt the source dataset first, rescaling its objects (*i.e.* bounding box dimensions & points in bounding boxes) to match size statistics in the target domain. Afterwards, they finetune the object detector on the adapted dataset. In short, SN attempts to explicitly construct a new size prior to replace the old one. Despite great effectiveness, SN is very sensitive to the accuracy of size statistics in the target domain, and requires careful retraining for each new target domain. In contrast, our work is a more general interpretation of size normalization, and instead rely on *shape invariance* to adapt it across all models and domains.

https://doi.org/10.52202/079017-3293

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: The paper's contributions and scope are accurately reflect in the the abstract and introduction.

   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: The limitations are discussed in the "Discussion and Future Work" section.

   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: [NA]

Justification: This paper does not include theory assumptions or proofs.

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: All the information needed to reproduce the main experimental results are clearly explained.

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?

 https://doi.org/10.52202/079017-3293

Answer: [Yes]

Justification: Code and instructions are included.

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, hyper-parameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: All the training and test details are specified.

   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: We did not test for statistical significance, as the large dataset size supports the reliability of the performance gains.

   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: The computer resources used are clearly explained.

   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 conform 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 impacts are discussed in the introduction section.

    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.

https://doi.org/10.52202/079017-3293

- 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: Assets are properly credited and license and terms are properly respected.

    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: [Yes]

    Justification: Code for reproduction is well documented.

    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.

https://doi.org/10.52202/079017-3293