# LiT: Unifying LiDAR "Languages" with LiDAR Translator


**Yixing Lao**
The University of Hong Kong
yxlao@cs.hku.hk

**Tao Tang**
Sun Yat-sen University
ttang@mail.sysu.edu.cn

**Xiaoyang Wu**
The University of Hong Kong
xywu@cs.hku.hk

**Peng Chen**
Cainiao Group
yuanshang.cp@cainiao.com

**Kaicheng Yu***
Westlake University
yukaicheng@westlake.edu.cn

**Hengshuang Zhao***
The University of Hong Kong
hszhao@cs.hku.hk



## Abstract

LiDAR data exhibits significant domain gaps due to variations in sensors, vehicles, and driving environments, creating "language barriers" that limit the effective use of data across domains and the scalability of LiDAR perception models. To address these challenges, we introduce the *LiDAR Translator (LiT)*, a framework that directly translates LiDAR data across domains, enabling both cross-domain adaptation and multi-domain joint learning. LiT integrates three key components: a scene modeling module for precise foreground and background reconstruction, a LiDAR modeling module that models LiDAR rays statistically and simulates ray-drop, and a fast, hardware-accelerated ray casting engine. LiT enables state-of-the-art zero-shot and unified domain detection across diverse LiDAR datasets, marking a step toward data-driven domain unification for autonomous driving systems. Source code and demos are available at: https://yxlao.github.io/lit.


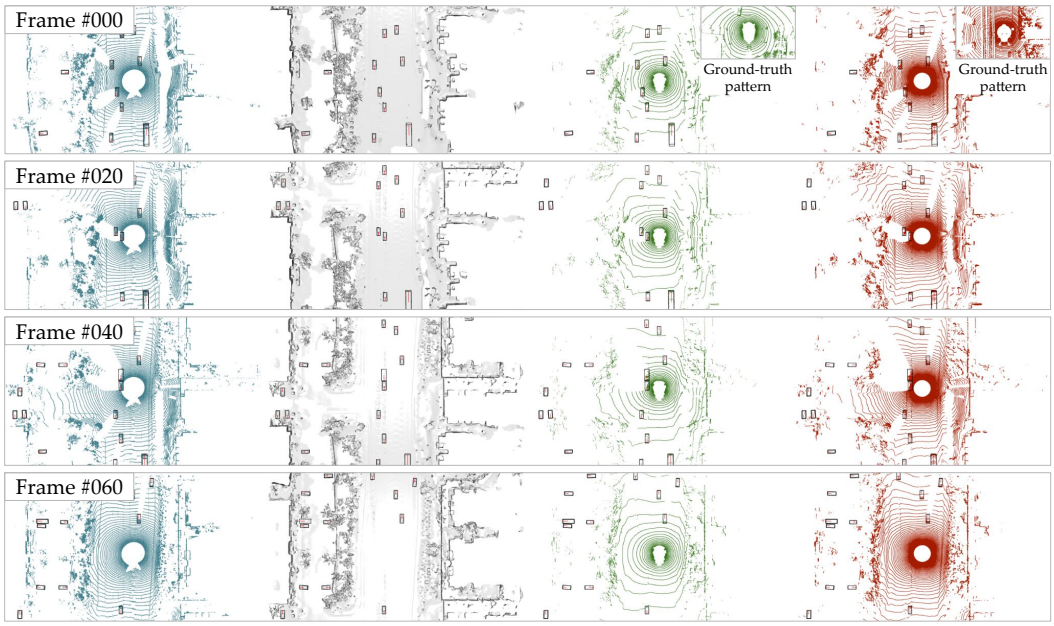


Figure 1: LiT translates LiDAR scenes across domains, capturing target domains' characteristics. By unifying the LiDAR "languages", LiT enables effective zero-shot and multi-dataset joint learning.


*Corresponding authors.



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

## 1 Introduction

LiDAR provides accurate geometry measurements of the surrounding environment, making it one of the most popular sensor choices for various autonomous systems [1, 2, 3, 4, 5]. Despite its importance, perception models trained on a specific LiDAR setup often struggle to generalize across different sensors [6, 7, 8], primarily due to the domain gaps attributable to variations in sensor intrinsic characteristics and driving environments. These gaps, akin to "language barriers," significantly degrade model performance when transferring between different LiDAR sensors and hinder effective joint training across multiple datasets. In a real-world scenario, it takes tremendous time and resources to collect, annotate, and train just one model for a specific LiDAR setup [9, 10, 11]. Consequently, this barrier acts as a bottleneck in training scale-up, which has been a driving force behind recent rapid advancements in other areas of representation learning, such as 2D vision [12, 13, 14, 15] and natural language processing (NLP) [16, 17, 18, 19]. Addressing the "language barriers" in the context of LiDAR data is therefore crucial for unlocking similar advancements within autonomous driving.

Recent efforts [6, 20] have already acknowledged the significance of "language barriers" within LiDAR data as a meaningful challenge. However, solutions to date primarily focus on model-driven approaches, attempting to bridge these gaps through adaptations within the perception model itself. While these methods have shown promise in bridging domain gaps in certain contexts, they introduce a notable drawback: the cost of customizing model structure and training data for new, specific domains. This requirement demands significant resources and limits the scalability and flexibility of deploying autonomous systems across varied environments.

To this end, we present the LiDAR Translator (LiT), a novel data-driven approach that functions similarly to a language translator, converting disparate LiDAR "dialects" into a common "language" (Fig. 1), and further unleashing the zero-shot capacity and joint training potential of LiDAR-based models. The overall pipeline is available in Fig. 4. Specifically, LiT reconstructs the source domain scene with neural implicit representations, capturing both the static background and dynamic foreground elements (Sec. 4.1). Then, LiT simulates the target domain LiDAR sensor model using a custom GPU-accelerated ray casting engine, generating translated LiDAR scans that faithfully replicate the target domain's characteristics (Sec. 4.2). By doing so, LiT effectively bridges the gap between different LiDAR "languages," enabling seamless domain adaptation and unification of disparate LiDAR datasets. Through this well-designed framework, LiT effectively bridges the gap between different LiDAR "languages," facilitating seamless domain unification of LiDAR datasets.

LiT advances domain adaptation through data-side unification and multi-domain joint learning [6]. We show in our experiments that, in single-source adaptation scenarios, LiT demonstrates superior zero-shot detection performance compared to existing model-driven and data-driven techniques. Additionally, by translating multiple data sources into a unified target domain, LiT facilitates multi-dataset joint training, where combining data from various sources not only improves model performance beyond single-source training but also establishes a new paradigm for data-centric learning strategies.

## 2 Related work

**3D representation learning.** In the realm of autonomous driving, achieving the unification of perception models for LiDAR data across different domains presents a distinct challenge, one that diverges from the significant progress observed in 2D domains where large-scale pre-training has dramatically enhanced downstream task performance [21]. Specifically in the domain of 3D representation learning, the field is still in an exploratory phase. Many existing studies have focused on training models from scratch, tailored to specific datasets [22]. Initially, this research largely targeted the recognition of individual objects [23, 24, 25, 26, 27]. Over time, it has evolved to address the more complex task of interpreting real-world, scene-centric point clouds [22, 28, 29, 30], marking a notable advance in our understanding of 3D data. Inspired by the achievements of large-scale representation learning and the effectiveness of multi-dataset synergistic approaches, such as Point Prompt Training (PPT) [6], LiT aims to overcome the constraints of model-side adaptations by utilizing a data-driven approach to bridging domain gaps.

**LiDAR domain adaptation.** LiDAR domain adaptation is critical for enabling models trained on one domain to effectively operate in another, accommodating variations such as sensor types and environmental conditions [31, 32, 33, 34, 35]. Techniques have varied widely: SN [36] tackles

Table 1: **Vanilla zero-shot capacity.** Without LiDAR translation, the performance of the 3D object detection model drops dramatically when applied to different domains. LiT is capable of translating LiDAR data across domains and can significantly improve the performance of the 3D object detection model. $AP_{BEV}$ and $AP_{3D}$ of the car category at IoU = 0.7 of the SECOND-IoU [50] model are shown.

| Method | Waymo → KITTI | | Waymo → nuScenes | | nuScenes → KITTI | |
|---|---|---|---|---|---|---|
| | $AP_{BEV}$ ↑ | $AP_{3D}$ ↑ | $AP_{BEV}$ ↑ | $AP_{3D}$ ↑ | $AP_{BEV}$ ↑ | $AP_{3D}$ ↑ |
| Without translation | 67.64 | 27.48 | 32.91 | 17.24 | 51.84 | 17.92 |
| **With LiT translation** | **82.55** | **69.94** | **37.00** | **22.19** | **80.54** | **60.13** |

object size discrepancies through normalization based on object statistics; ST3D [7] and ST3D++ [8] utilize a self-training pipeline with pseudo-labels for fine-tuning; LiDAR-Distillation [37] mitigates beam-induced domain shifts with a progressive framework; SPG [38] aims to fill in missing points in foreground areas; and 3D-CoCo [39] employs domain-specific encoders and contrastive learning for transferable representation. LiT aligns with these efforts by employing zero-shot object detection as a primary benchmark to measure its effectiveness in domain adaptation. However, LiT's ambition extends beyond mere adaptation; it aims to unify the "languages" of multiple LiDAR domains, exploring the synergistic potential of such unified data.

**Autonomous driving simulator.** Simulators play an important role in autonomous driving research by making data collection and annotation much easier and cheaper. Initial efforts leveraged ground-truth information extracted from graphic engines, such as CARLA [40], by utilizing various simulators [41, 42, 43, 44, 45]. Despite their utility, these sim-to-real approaches often involve complex dependencies on simulator engines, which can be cumbersome and costly to establish, and still result in a notable sim-to-real gap. In response to these challenges, subsequent studies like LiDARsim [46] and PCGen [47] have attempted to bridge this gap by simulating point clouds from real data through multi-step, data-driven pipelines. More recent advances, including UniSim [48] and LiDAR-NeRF [49], have explored the use of neural implicit fields to simulate new sensor views. However, these methods primarily generate data reflective of the source domain and do not adequately address variations across datasets. Different from these works, LiT enables direct translation of real-world source data into target domain LiDAR characteristics, effectively bridging the domain gap through simulation.

## 3 Pilot study

### 3.1 Exploring language barriers among LiDARs

The progression of autonomous driving technologies is intimately tied to the diversity and quality of LiDAR data, which provides a detailed three-dimensional representation of environments. Operating in varied settings, autonomous vehicles utilize diverse LiDAR sensors, each characterized by unique specifications like field of view, beam count, and ray-drop behavior. This results in notable variations across datasets such as Waymo [10], nuScenes [11], and KITTI [9], creating significant domain gaps akin to language barriers between LiDAR "languages".

These gaps manifest in various forms, including alterations in point cloud density due to beam count differences and changes in spatial coverage from FoV variations. Additionally, scene content differences (such as vehicle sizes, orientations, and types) mirror the geographical and operational diversity of each dataset, adding layers of complexity to the domain-specific challenges (as visualized in Fig. 2). Such disparities obstruct the direct application of models trained on one dataset to another (see Table 1) and further pose significant obstacles to exploring synergistic training across these datasets. By tackling these barriers head-on, LiT aims to unlock the potential of data scale.

*We wish to break the language barriers among LiDARs with our LiDAR Translator.*

### 3.2 Embracing data-driven domain unification

Previous attempts to bridge domain gaps [7, 8], particularly through unsupervised domain adaptation techniques, have achieved limited success. These model-based approaches focus on adapting the model to fit the target domain, often neglecting the underlying variations in data characteristics. As a

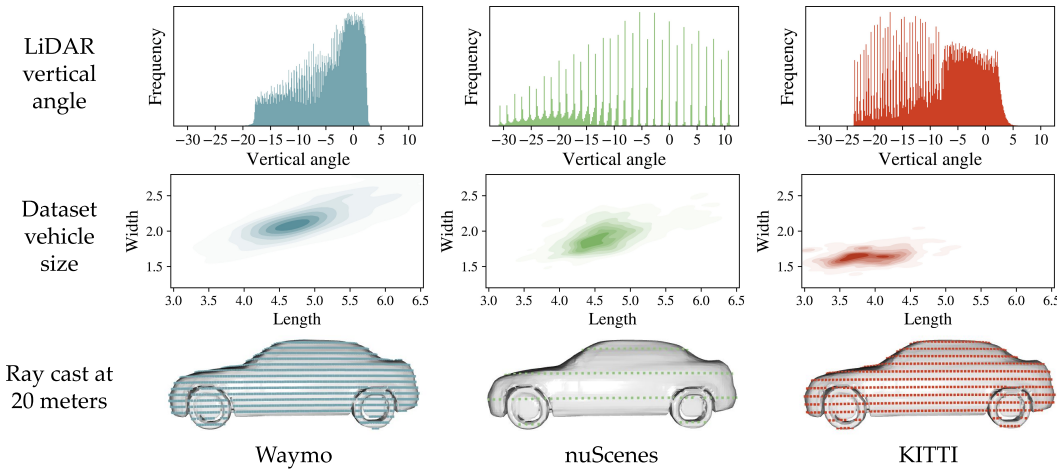


Figure 2: **Domain gaps for LiDARs.** *Top Row:* LiDAR ray angles have significantly different distributions. We model their statistical distributions as detailed in Sec. 4.2. *Middle Row:* Foreground vehicle sizes can differ across datasets. *Bottom Row:* We show the ideal ray casting results from LiDARs mounted at 1.6 meters height to a reconstructed vehicle placed 20 meters in front.

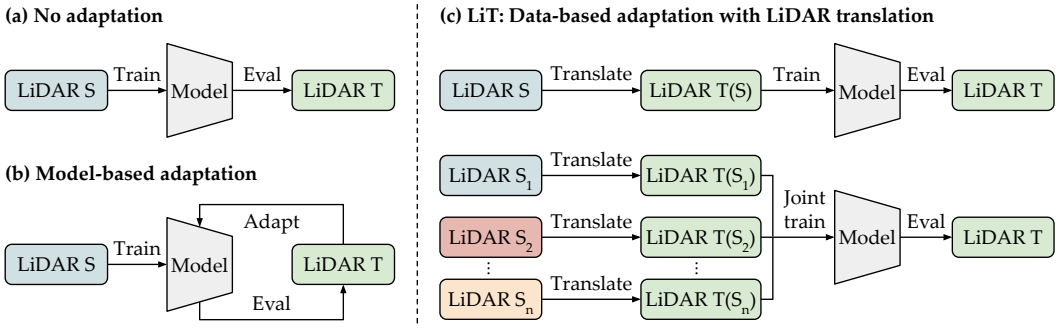


Figure 3: **Comparing LiT with model-based adaptation.** (a) Training a model on source domain data and directly applying it to the target domain typically results in poor performance due to the domain gap. (b) Model-based adaptation techniques [7] adapt the model to the target domain but do not explicitly model the target domain data LiDAR characteristics and data distribution. (c) LiT directly translates LiDAR data from the multiple source domains to a unified target domain, effectively bridging the domain gaps and enabling joint training across multiple datasets.

result, while they may narrow the domain gap to a degree, they do not fully exploit the potential of diverse datasets to bolster model robustness and generalization.

Contrary to model-side adaptation, data-driven adaptation presents a more practical and efficient avenue for LiDAR domain unification without the need for additional adjustment on the base model. By translating LiDAR data from various source domains into a standardized target domain "language", we can unify discrepancies across datasets. This harmonization creates a cohesive dataset that reflects the diversity of multiple sources, addressing the issue of negative transfer and enriching the training data to boost model performance and generalizability in different settings.

Specifically, if we can translate data from an available source domain to an unseen target domain, we can further use the translated data for model training and fulfill downstream tasks on the target domain, effectively achieving zero-shot cross-domain learning. Moreover, if we can unify multiple source domains into one target domain, model training can further take advantage of a larger scale of data training to push the model's generalization capabilities to a higher level, as illustrated in Fig. 3. In a word, embracing data-driven domain unification has great potential in both zero-shot adaptations from source to target and large-scale multi-domain joint training.

*We treat data-driven domain unification as the original point of our LiDAR Translator.*

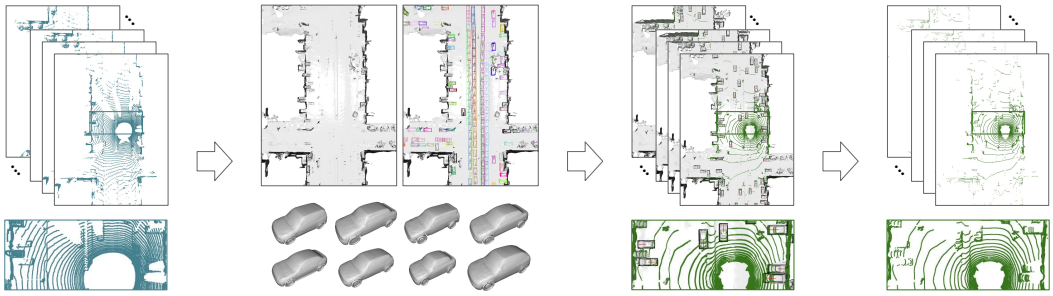


Figure 4: **LiT pipeline overview.** LiT translates LiDAR data across domains, integrating scene modeling (Sec. 4.1) and LiDAR modeling (Sec. 4.2) with GPU-accelerated ray casting. LiT is highly efficient, as it can translate a multi-frame LiDAR scene in typically less than one minute (Table 6).

# 4 LiDAR translator

## 4.1 Scene modeling

**Foreground modeling.** Previous work ReSimAD [51] relies on hand-crafted 3D vehicle assets from simulator engines [40], which limits the diversity of the foreground objects and the realism of the simulated data. In contrast, LiT reconstructs the foreground objects from multi-frame LiDAR point clouds, which are more representative of real-world data with diverse vehicle geometries. To achieve this, we first track the vehicles across multiple LiDAR frames, where the tracking information is typically provided by LiDAR datasets [9, 10, 11]. Inspired by the recent success of implicit neural representations [52, 53] in reconstructing shapes from LiDAR point clouds, we train an SDF-based reconstruction model on the ShapeNetV2 [54] vehicle category and apply it to reconstruct the foreground objects from multi-frame LiDAR point clouds. With this approach, we can accurately reconstruct the detailed geometry of diverse vehicle shapes. As shown in our experiments in Table 5, the increase in foreground diversity leads to a significant performance increase in zero-shot detection tasks. We refer the reader to Sec. A.1 for details and visualizations on the foreground modeling.

**Background modeling.** The primary goal of background modeling is to faithfully reconstruct the background environment. However, the efficiency and scalability of the background modeling method are equally important, especially when dealing with joint training on multiple large-scale datasets. ReSimAD [51] uses NeuS [55] for background reconstruction, which requires custom training for each scene with time measured in hours. In contrast, LiT models the scene as the zero-level set of a 3D implicit field defined by a hierarchical neural kernel field [56, 57]. This approach is generalizable, eliminating the need for retraining on every new LiDAR sequence and significantly reducing LiT's full-scene translation time to less than a minute for a 200-frame LiDAR scene, as detailed in Table 6. This efficiency plays a crucial role in enabling the scalability of LiT across various datasets. We refer readers to Sec. A.2 in the supplementary material for further details and visualizations on background modeling.

**Module adaptability.** Both the foreground and background modeling are crucial modules for achieving high fidelity in scene representation, enabling accurate domain adaptation. In LiT, these modules are designed to be swappable, allowing for the integration of more advanced scene representation methods as they are developed. While the specific modeling techniques are important, our primary contribution lies in how these components work together within our domain translation framework to enable effective domain unification.

## 4.2 LiDAR modeling

Modeling the different LiDAR characteristics is a critical step in domain adaptation, as it directly affects the simulated data distribution and characteristics of the target domain. Unlike existing work [51] that relies on full-scale simulators, we develop a custom LiDAR modeling and ray casting pipeline that is more flexible and efficient, and it does not require complex dependencies when using simulator engines. We model the LiDAR characteristics of beam angles. All of these are made possible by our hardware-accelerated LiDAR ray casting engine.

**Statistical modeling of LiDAR ray angles.** The naive implementation for LiDAR ray generation is to evenly distribute the rays in the horizontal axis, based on the LiDAR's horizontal field of view (FOV) and beam counts. However, this approach does not accurately reflect the real-world LiDAR characteristics, which are often non-homogeneous in the vertical direction, as shown in the top row of Fig. 2. To address this issue, we conduct a statistical analysis of LiDAR rays, focusing on the ray distribution in the vertical axis. First, for each LiDAR ray in the dataset, we compute its vertical angle based on the ray's xy-plane distance and z-axis distance. Then, we calculate the distribution of the vertical angles and identify the peak angles where the majority of the LiDAR rays are concentrated. Specifically, this is done by finding $H$ statistical modes in the distribution of vertical angles, where $H$ is the height of the range image or the number of beams in the LiDAR sensor. Notably, we do not require manual labeling of target domain data, only a small amount of target domain LiDAR data to perform this statistical analysis. After the modeling, our LiDAR ray generation module then generates rays based on these identified peak angles, ensuring that our simulated LiDAR data is representative of real-world LiDAR characteristics. Fig. 5 shows the average Chamfer distance between the real-world LiDAR data and the self-to-self translated LiDAR data on nuScenes. The chamfer distance is reduced when statistical modeling is applied (blue curve) compared to the vanilla case (red curve), indicating that the simulated LiDAR data more closely resemble the real-world LiDAR data.

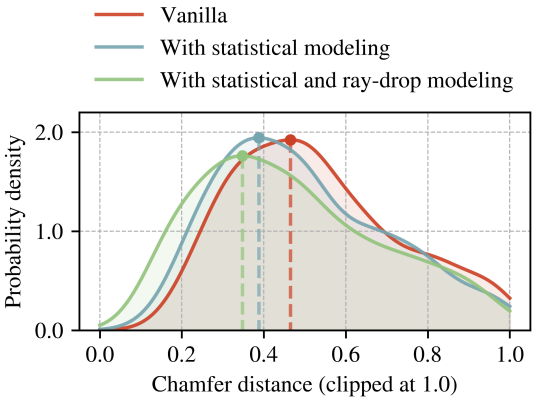


Figure 5: **LiDAR modeling with statistical ray angles and ray-drop.** We compare the chamfer distance between the real-world LiDAR data and the simulated LiDAR data on nuScenes. The chamfer distance is smallest when both statistical modeling and ray-drop modeling are applied. The chamfer distances are clipped at 1.0m for visualization.

**LiDAR ray-drop modeling.** Ray-drop of LiDAR rays is frequent in real-world scenarios due to various factors, including object reflectivity, absorption, or occlusion. We model this phenomenon by training a ray-drop predictor model that predicts the likelihood of a ray being dropped. To collect the ray-drop dataset, we reconstruct background and foreground scenes and perform ray casting using the dataset's own LiDAR parameters. For each ray, we record whether it hits a mesh or not in the simulated scene. Inspired by PCGen [47], we collect the ray direction $\mathbf{r}$, the point distance $d$, and the ray incident angle $\theta$ for each ray. Then, we train an MLP-based ray-drop model with positional encoding [58], to predict the likelihood of a ray being dropped. The MLP network takes $(\mathbf{r}, d, \theta)$ with positional encoding as input, and outputs the probability of the ray being dropped.

For quantitative evaluation, we perform a self-to-self translation task on the nuScenes dataset to assess the quality of the ray-drop modeling. Fig. 5 shows the chamfer distance decreases when ray-drop modeling is applied, and it shows the combined effects of applying both statistical modeling and ray-drop modeling together. For qualitative evaluation, we present the visualization of the ray-drop model trained on the nuScenes dataset in Fig. 6. The simulated LiDAR points, especially those proximal to vehicles, effectively resemble the nuScenes dataset's characteristic patterns.

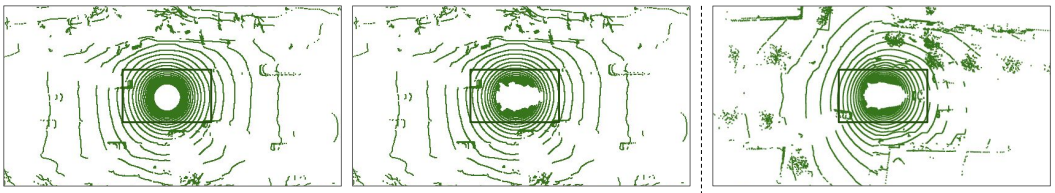


Figure 6: **Ray-drop modeling visualization.** The left image shows the translated nuScenes frame without ray-drop modeling, where it has a dense circular LiDAR pattern near the vehicle. The middle image, with ray-drop modeling applied, displays sparser LiDAR points near the vehicle, closely resembling nuScenes' scan patterns. The right image is a real nuScenes LiDAR scan from another scene.

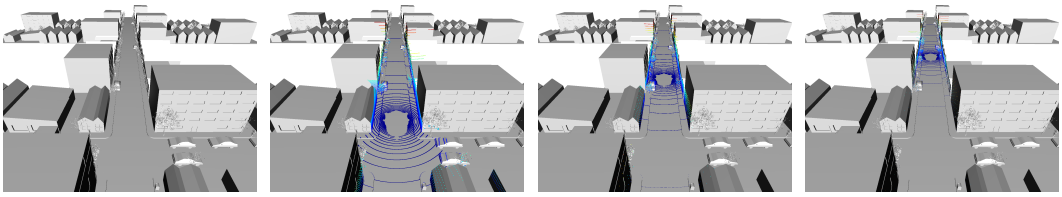

Figure 7: **Deploying LiT-modeled LiDAR to a new scene.** The LiT pipeline allows flexible composition of LiDARs and scenes, even when the scene is not modeled by LiT. We show visualizations of the synthetic Mai City scene [59] (column 1), and the LiT-simulated LiDAR scans in nuScenes patterns following a moving vehicle trajectory (column 2-4).

**Hardware accelerated LiDAR ray casting.** While existing work [51] relies on integrating with full-scale simulators like CARLA [40], which can be cumbersome to recompile and less efficient to run, we have developed a hardware-accelerated LiDAR ray casting engine for LiT that offers greater flexibility and efficiency. Specifically, our LiDAR ray casting engine emits rays based on the LiDAR modeling, computes the location of the ray-mesh hit point, the surface normal at that point, and the incident angle of the ray, providing the necessary data to simulate target domain points. Given the surface normal $\mathbf{n}$ of the triangle hit by the ray and the ray direction $\mathbf{r}$, the incident angle $\theta$ is computed as $\theta = \arccos\left((\mathbf{r} \cdot \mathbf{n})/(\|\mathbf{r}\|\|\mathbf{n}\|)\right)$. LiT's ray casting engine is accelerated on both CPU with Intel Embree and GPU with Nvidia OptiX, enabling real-time ray casting (22Hz to 31Hz) when running on a single GPU. The efficiency of this engine allows LiT to translate a multi-frame LiDAR scene typically in less than one minute, making large-scale domain unification tasks feasible. For more detailed runtime statistics, refer to Table 6.

**Flexible composition of LiDARs and scenes.** LiT's flexible pipeline allows seamless deployment of LiT-modeled LiDARs to new environments that are not necessarily modeled by LiT. As demonstrated in Fig. 7, we can deploy a LiT-modeled LiDAR in nuScenes's style in a given new environment, such as the Mai City [59] scene. This flexibility allows LiT to work with various sources of reconstructed 3D maps, while being able to simulate LiDAR scans with target LiDAR sensor characteristics, which is particularly useful for deploying new LiDARs or simulating out-of-distribution scenarios.

# 5 Experiment

## 5.1 Experimental settings

**Datasets.** Our experiments are conducted on three widely utilized datasets for autonomous driving research: KITTI [9], Waymo [10], and nuScenes [11]. These datasets present a diverse range of LiDAR characteristics, including varying beam counts and sensor configurations, making them ideal for evaluating our LiDAR translation approach, LiT. Adaptation scenarios covered include cross-beam-count adaptations such as Waymo $\rightarrow$ nuScenes (high-beam to low-beam) and nuScenes $\rightarrow$ KITTI (low-beam to high-beam), as well as adaptations within the same beam count but different LiDAR characteristics like Waymo $\rightarrow$ KITTI. Table 6 provides an overview of the dataset statistics, including details on LiDAR beams and the distribution of foreground objects.

**Evaluation metrics.** To evaluate how well our translated point clouds match the statistical distribution of the target domain, we follow LiDARGen [60] and UltraLiDAR [61] to compute Maximum-Mean Discrepancy (MMD) and Jensen-Shannon divergence (JSD) in BEV voxel occupancy between the source domain point clouds translated to the target domain style and the actual target domain point clouds. As the source and target domains come from entirely different scenes, we measure the overall distributional similarity rather than per-scene alignment, providing a quantitative measure of how well LiT captures the general characteristics of the target domain.

The downstream detection task evaluation follows the KITTI benchmark standards, focusing on the car category. We report Average Precision (AP) across 40 recall positions at an Intersection over Union (IoU) threshold of 0.7 for both bird's eye view (BEV) and 3D detection. For KITTI, metrics are reported for front-view annotations only, while for Waymo and nuScenes, we extend our evaluation to include full ring-view point clouds generated from top-mounted LiDAR sensors, while other LiDARs

Table 2: **Statistical alignment with target domain.** We report the distributional differences between translated and ground-truth target domains with Maximum-Mean Discrepancy (MMD) and Jensen-Shannon divergence (JSD). Baseline represents LiDAR data from the source domain.

| Task | Beam count | $\text{MMD}_{\text{BEV}} \downarrow$ | | $\text{JSD}_{\text{BEV}} \downarrow$ | |
|---|---|---|---|---|---|
| | | Baseline | **LiT** | Baseline | **LiT** |
| Waymo → KITTI | 64-beam → 64-beam | 8.817e-04 | **3.268e-04** | 0.273 | **0.180** |
| Waymo → nuScenes | 64-beam → 32-beam | 2.310e-03 | **6.583e-04** | 0.380 | **0.205** |
| nuScenes → KITTI | 32-beam → 64-beam | 8.725e-04 | **2.107e-04** | 0.220 | **0.164** |

Table 3: **Single source domain unification.** We compare the $\text{AP}_{\text{BEV}}$ and $\text{AP}_{\text{3D}}$ of the car category at IoU = 0.7 as well as the domain gap closed by different methods. Source only denotes that the pre-trained detector is directly evaluated on the target domain, and Oracle represents the detection results trained on the fully annotated target domain. We highlight the best results in **bold**.

| Task | Method | SECOND-IoU [50] | | PV-RCNN [62] | |
|---|---|---|---|---|---|
| | | $\text{AP}_{\text{BEV}} \uparrow$ / $\text{AP}_{\text{3D}} \uparrow$ | Closed gap ↑ | $\text{AP}_{\text{BEV}} \uparrow$ / $\text{AP}_{\text{3D}} \uparrow$ | Closed gap ↑ |
| Waymo → KITTI | Source only | 67.64 / 27.48 | – | 61.18 / 22.01 | – |
| | SN [36] | 78.96 / 59.20 | +72.33% / +69.00% | 79.78 / 63.60 | +66.91% / +68.76% |
| | ST3D [7] | 82.19 / 61.83 | +92.97% / +74.72% | 84.10 / 64.78 | +82.45% / +70.71% |
| | ReSimAD [51] | – | – | 81.01 / 58.42 | +71.33% / +60.19% |
| | **LiT** | **82.55 / 69.94** | +95.27% / +92.36% | **84.35 / 65.68** | +83.35% / +72.19% |
| | Oracle | 83.29 / 73.45 | – | 88.98 / 82.50 | – |
| Waymo → nuScenes | Source only | 32.91 / 17.24 | – | 34.50 / 21.47 | – |
| | SN [36] | 33.23 / 18.57 | +01.69% / +07.54% | 34.22 / 22.29 | -01.50% / +04.80% |
| | ST3D [7] | 35.92 / 20.19 | +15.87% / +16.73% | 36.42 / 22.99 | +10.32% / +08.89% |
| | ReSimAD [51] | – | – | 37.85 / 21.33 | +18.00% / -00.81% |
| | **LiT** | **37.00 / 22.19** | +21.56% / +28.08% | **38.77 / 23.48** | +22.94% / +11.76% |
| | Oracle | 51.88 / 34.87 | – | 53.11 / 38.56 | – |
| nuScenes → KITTI | Source only | 51.84 / 17.92 | – | 68.15 / 37.17 | – |
| | SN [36] | 40.03 / 21.23 | -37.55% / +05.96% | 60.48 / 49.47 | -36.82% / +27.13% |
| | ST3D [7] | 75.94 / 54.13 | +76.63% / +59.50% | 78.36 / 70.85 | +49.02% / +74.30% |
| | ReSimAD [51] | – | – | – | – |
| | **LiT** | **80.54 / 60.13** | +91.26% / +76.01% | **85.82 / 74.87** | +84.83% / +83.17% |
| | Oracle | 83.29 / 73.45 | – | 88.98 / 82.50 | – |

are not used in our experiment. The evaluations are conducted using the SECOND-IoU [50] and PV-RCNN [62] models, following the evaluation protocol of previous works [36, 7, 51].

**Baselines and comparison.** For the domain unification tasks, LiT is evaluated against key baselines in autonomous driving domain adaptation: (i) *SN* [36], which normalizes object sizes using target domain statistics; (ii) *ST3D* [7], employing iterative pseudo label generation and curriculum-based training; (iii) *ReSimAD* [51], a method combining reconstruction and simulation; and (iv) *Oracle*, the theoretical performance upper bound via full target domain supervision.

### 5.2 Experimental results

**Statistical alignment with target domain.** To validate the effectiveness of our LiDAR translation approach directly, we compare the distributional differences between translated and ground-truth target domains, as shown in Table 2. Across all adaptation scenarios, LiT significantly reduces both MMD and JSD metrics compared to the baseline, which is the LiDAR data from the source domain. More substantial improvements are observed in cross-beam-count scenarios (64-beam → 32-beam and 32-beam → 64-beam), where LiT achieves $3 \sim 4\times$ reduction in $\text{MMD}_{\text{BEV}}$.

**Single source domain unification.** In the single source domain adaptation setting as shown in Table 3, LiT consistently outperforms all baselines, including the model-based adaptation approach ST3D and data-based adaptation approach ReSimAD, in both the $\text{AP}_{\text{BEV}}$ and $\text{AP}_{\text{3D}}$ metrics across different settings. Notably, in the challenging nuScenes → KITTI tasks, where the low-beam LiDAR

Table 4: **Multi source domain unification.** We compare models trained with one, two, and three source domains. In the table, W = Waymo, N = nuScenes, and K = KITTI. We report the $AP_{BEV}$ and $AP_{3D}$ of the car category at IoU = 0.7. The last row represents the "oracle" K $\rightarrow$ K model trained with full target-domain supervision. Remarkably, in the W + N $\rightarrow$ K task, LiT surpasses the oracle performance in SECOND-IoU's $AP_{BEV}$ even when the target domain is never seen during training. When all three source domains are used, LiT surpasses the oracle performance in all metrics.

| Task | Method | Zero-shot | SECOND-IoU [50] | | PV-RCNN [62] | |
|---|---|---|---|---|---|---|
| | | | $AP_{BEV}$ ↑ / $AP_{3D}$ ↑ | Closed gap ↑ | $AP_{BEV}$ ↑ / $AP_{3D}$ ↑ | Closed gap ↑ |
| W $\rightarrow$ K | Source Only | Yes | 67.64 / 27.48 | – | 61.18 / 22.01 | – |
| | **LiT** | Yes | **82.55 / 69.94** | +95.27% / +92.36% | **84.35 / 65.68** | +83.35% / +72.19% |
| N $\rightarrow$ K | Source Only | Yes | 51.84 / 17.92 | – | 68.15 / 37.17 | – |
| | **LiT** | Yes | **80.54 / 60.13** | +91.26% / +76.01% | **85.82 / 74.87** | +84.83% / +83.17% |
| W + N $\rightarrow$ K | Source Only | Yes | 67.26 / 22.05 | – | 77.82 / 34.05 | – |
| | **LiT** | Yes | **84.45 / 71.58** | +107.24% / +96.36% | **84.15 / 75.50** | +56.72% / +85.55% |
| W + N + K $\rightarrow$ K | **LiT** | No | **87.52 / 75.76** | – | **90.67 / 82.67** | – |
| K $\rightarrow$ K | Oracle | No | 83.29 / 73.45 | – | 88.98 / 82.50 | – |

Table 5: **Ablation studies.** We report the performance of LiT with nuScenes $\rightarrow$ KITTI translation tasks with SECOND-IoU [50] model to study the effects of different configurations in LiT.

| Group | Setting | $AP_{BEV}$ ↑ | $AP_{3D}$ ↑ |
|---|---|---|---|
| Source only | – | 51.84 | 17.92 |
| Foreground diversity | Shared 1 foreground mesh | 76.51 | 55.30 |
| | Shared 50 foreground mesh | 77.23 | 57.27 |
| | Foreground simulation only | 80.14 | 48.45 |
| Foreground inaccuracies | Noise std = 0.01m | 77.14 | 57.84 |
| | Noise std = 0.02m | 77.25 | 56.78 |
| | Noise std = 0.05m | 69.57 | 35.37 |
| Background inaccuracies | Noise std = 0.01m | 78.02 | 61.43 |
| | Noise std = 0.02m | 78.70 | 57.99 |
| | Noise std = 0.05m | 76.76 | 58.21 |
| Foreground and background inaccuracies | Noise std = 0.01m | 77.54 | 59.60 |
| | Noise std = 0.02m | 76.90 | 56.09 |
| | Noise std = 0.05m | 72.03 | 36.18 |
| **LiT full model** | – | **80.54** | **60.13** |

data from nuScenes is adapted to the high-beam LiDAR data from KITTI, LiT achieves significant improvements over ST3D, closing the gap of $AP_{BEV}$ from 76.63% to 91.26% and 49.02% to 84.83% for SECOND-IOU and PV-RCNN, respectively.

**Multi source domain unification.** As shown in Table 4, the performance of LiT improves further when trained with multiple source domains, highlighting the benefits of being able to leverage diverse data sources via LiDAR translation. In a zero-shot setup, combining Waymo and nuScenes training sets naively (source only) does not improve the performance much or may harm the performance in some scenarios. After LiT translation, the combined Waymo nuScenes training set achieves much better performance compared to the naive combination and single source domain training. Remarkably, LiT surpasses the oracle performance (achieving $AP_{BEV}$ of 84.45 over 83.29) in SECOND-IoU under the Waymo + nuScenes $\rightarrow$ KITTI adaptation task, demonstrating its effectiveness in zero-shot scenario when the target domain is never seen during training. When all three source domains are used, LiT achieves the best performance, surpassing the oracle performance in all metrics.

**Foreground diversity.** The diversity of foreground objects plays an important role in the domain adaptation's performance. Specifically, when transitioning from a single shared foreground mesh to 50 shared meshes sees an improvement in both $AP_{BEV}$ and $AP_{3D}$. Furthermore, with only foreground simulation, LiT still manages to achieve reasonable performance, showcasing LiT's robustness. The results are shown in Table 5.

Table 6: **Full-scene LiDAR translation in under 1 minute.** We present averaged runtime and key statistics for the LiT LiDAR translation pipeline. Here, a "scene" is a LiDAR sequence containing multiple LiDAR point cloud "frames". Runtime is measured on a single NVIDIA RTX 4090 GPU.

| Pipeline | Item | Waymo → KITTI | Waymo → nuScenes | nuScenes → KITTI |
|---|---|---|---|---|
| Background modeling | # Frames per scene | 198.07 | 198.07 | 40.26 |
| | # Points per frame | 146,580.51 | 146,580.51 | 23,988.50 |
| | # Points per scene | 29,033,201.66 | 29,033,201.66 | 965,776.97 |
| | # Vertices of recon mesh | 1,843,176.99 | 1,843,176.99 | 1,200,037.31 |
| | Recon. time per frame | 0.11 Sec. | 0.11 Sec. | 0.16 Sec. |
| | Recon. time per scene | 21.97 Sec. | 21.97 Sec. | 6.51 Sec. |
| Foreground modeling | # Vehicles per scene | 45.20 | 45.20 | 49.15 |
| | # LiDAR frames per vehicle | 79.03 | 79.03 | 11.42 |
| | # Multi-Frame points per vehicle | 40,506.01 | 40,506.01 | 1,089.57 |
| | Recon. time per scene | 24.74 Sec. | 24.74 Sec. | 29.04 Sec. |
| Ray casting | # Emitted LiDAR rays per frame | 119,232.00 | 34,880.00 | 119,232.00 |
| | # Ray hit per frame | 115,937.99 | 29,834.54 | 116,058.03 |
| | Ray casting time per frame | 0.04 Sec. | 0.03 Sec. | 0.03 Sec. |
| | Ray casting time per scene | 8.89 Sec. | 6.09 Sec. | 1.27 Sec. |
| LiT translation time per scene (all frames) | | 55.60 Sec. | 52.80 Sec. | 36.82 Sec. |

**Scene modeling inaccuracies.** To understand how modeling accuracy affects performance, we simulate inaccuracies by adding Gaussian noise with $\mathrm{std} = 0.01\mathrm{m}$, $0.02\mathrm{m}$, and $0.05\mathrm{m}$ to the reconstructed mesh vertices of foreground-only, background-only, and both foreground and background. As shown in Table 5, the detection model is more sensitive to inaccuracies in the foreground compared to background, which is expected as foreground objects are the main target for object detection. Even with scene modeling inaccuracies, LiT still substantially outperforms the source-only (no translation) baseline, demonstrating the robustness of our approach.

**Runtime performance.** Table 6 shows the runtime performance and key statistics of LiT's LiDAR translation process. We report the average runtime and data statistics of 50 scenes from each dataset. The runtime is measured on a desktop PC with a single NVIDIA RTX 4090 GPU. Notably, LiT is capable of translating a full multi-frame LiDAR scene from one domain to another in under a minute. After the reconstruction steps are done, the LiDAR ray casting process is highly efficient and can run in real-time. The efficiency of LiT demonstrates its scalability in real-world applications, where the ability to quickly adapt sensor data to different domains can enhance perception systems.

## 6 Conclusion

In conclusion, we present the LiDAR Translator (LiT), a pioneering framework that unifies LiDAR data into a common "language" in a comprehensive system consisting of scene modeling, LiDAR modeling, and domain adaptation. LiT overcomes domain discrepancies in LiDAR data from diverse sensor setups and environments, enabling seamless integration of multiple LiDAR datasets, enhancing zero-shot detection capabilities, and improving the representation quality of pre-trained models across a variety of scenarios. Additionally, LiT's efficiency streamlines data preprocessing, reducing both time and computational demands, and facilitating quicker development cycles in autonomous systems. This acceleration promotes rapid advancements and the deployment of safer, more effective vehicles.

**Social impact and limitations.** The LiT framework significantly advances autonomous driving by facilitating domain adaptation for LiDAR data, which could enhance transportation safety and efficiency. However, it has limitations, such as its exclusive reliance on LiDAR data, its dependency on annotated source datasets for foreground reconstruction, and its current focus on vehicle objects solely. Future enhancements should aim to incorporate other object types and sensor modalities, as well as reduce dependency on extensive labeling, to broaden LiT's applicability and impact.

**Acknowledgement.** This work is supported by the National Natural Science Foundation of China (No. 62201484), Alibaba Innovative Research Fund, HKU Startup Fund, and HKU Seed Fund for Basic Research.

## References


[1] Alex H Lang, Sourabh Vora, Holger Caesar, Lubing Zhou, Jiong Yang, and Oscar Beijbom. Pointpillars: Fast encoders for object detection from point clouds. In *CVPR*, 2019. 2

[2] Tianwei Yin, Xingyi Zhou, and Philipp Krähenbühl. Center-based 3d object detection and tracking. In *CVPR*, 2021. 2

[3] Lue Fan, Ziqi Pang, Tianyuan Zhang, Yu-Xiong Wang, Hang Zhao, Feng Wang, Naiyan Wang, and Zhaoxiang Zhang. Embracing single stride 3d object detector with sparse transformer. In *CVPR*, 2022. 2

[4] Chao Ma Guangsheng Shi, Ruifeng Li. Pillarnet: Real-time and high-performance pillar-based 3d object detection. In *ECCV*, 2022. 2

[5] Pointcept Contributors. Pointcept: A codebase for point cloud perception research. https://github.com/Pointcept/Pointcept, 2023. 2

[6] Xiaoyang Wu, Zhuotao Tian, Xin Wen, Bohao Peng, Xihui Liu, Kaicheng Yu, and Hengshuang Zhao. Towards large-scale 3d representation learning with multi-dataset point prompt training. In *CVPR*, 2024. 2

[7] Jihan Yang, Shaoshuai Shi, Zhe Wang, Hongsheng Li, and Xiaojuan Qi. ST3D: self-training for unsupervised domain adaptation on 3d object detection. In *CVPR*, 2021. 2, 3, 4, 8

[8] Jihan Yang, Shaoshuai Shi, Zhe Wang, Hongsheng Li, and Xiaojuan Qi. St3d++: Denoised self-training for unsupervised domain adaptation on 3d object detection. *PAMI*, 2022. 2, 3

[9] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. Vision meets robotics: The KITTI dataset. *The International Journal of Robotics Research*, 2013. 2, 3, 5, 7, 14

[10] Pei Sun, Henrik Kretzschmar, Xerxes Dotiwalla, Aurelien Chouard, Vijaysai Patnaik, Paul Tsui, James Guo, Yin Zhou, Yuning Chai, Benjamin Caine, Vijay Vasudevan, Wei Han, Jiquan Ngiam, Hang Zhao, Aleksei Timofeev, Scott Ettinger, Maxim Krivokon, Amy Gao, Aditya Joshi, Yu Zhang, Jonathon Shlens, Zhifeng Chen, and Dragomir Anguelov. Scalability in perception for autonomous driving: Waymo open dataset. In *CVPR*, 2020. 2, 3, 5, 7, 14

[11] 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. 2, 3, 5, 7, 14

[12] Yonglong Tian, Olivier J Henaff, and Aäron van den Oord. Divide and contrast: Self-supervised learning from uncurated data. In *CVPR*, 2021. 2

[13] Priya Goyal, Mathilde Caron, Benjamin Lefaudeux, Min Xu, Pengchao Wang, Vivek Pai, Mannat Singh, Vitaliy Liptchinsky, Ishan Misra, Armand Joulin, et al. Self-supervised pretraining of visual features in the wild. *arXiv:2103.01988*, 2021. 2

[14] Xinlong Wang, Wen Wang, Yue Cao, Chunhua Shen, and Tiejun Huang. Images speak in images: A generalist painter for in-context visual learning. In *CVPR*, 2023. 2

[15] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C. Berg, Wan-Yen Lo, Piotr Dollár, and Ross Girshick. Segment anything. In *ICCV*, 2023. 2

[16] Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *arXiv:2001.08361*, 2020. 2

[17] Vamsi Aribandi, Yi Tay, Tal Schuster, Jinfeng Rao, Huaixiu Steven Zheng, Sanket Vaibhav Mehta, Honglei Zhuang, Vinh Q. Tran, Dara Bahri, Jianmo Ni, Jai Gupta, Kai Hui, Sebastian Ruder, and Donald Metzler. Ext5: Towards extreme multi-task scaling for transfer learning. In *ICLR*, 2022. 2

[18] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. Llama: Open and efficient foundation language models. *arXiv:2302.13971*, 2023. 2

[19] OpenAI. Gpt-4 technical report. *arXiv:2303.08774*, 2023. 2

[20] Bo Zhang, Jiakang Yuan, Botian Shi, Tao Chen, Yikang Li, and Yu Qiao. Uni3d: A unified baseline for multi-dataset 3d object detection. In *CVPR*, 2023. 2

[21] Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers. In *CVPR*, 2021. 2

[22] Saining Xie, Jiatao Gu, Demi Guo, Charles R Qi, Leonidas Guibas, and Or Litany. Pointcontrast: Unsupervised pre-training for 3d point cloud understanding. In *ECCV*, 2020. 2

[23] Yue Wang and Justin M Solomon. Deep closest point: Learning representations for point cloud registration. In *ICCV*, 2019. 2

[24] Jonathan Sauder and Bjarne Sievers. Self-supervised deep learning on point clouds by reconstructing space. In *NeurIPS*, 2019. 2

[25] Aditya Sanghi. Info3d: Representation learning on 3d objects using mutual information maximization and contrastive learning. In *ECCV*, 2020. 2

[26] Yatian Pang, Wenxiao Wang, Francis EH Tay, Wei Liu, Yonghong Tian, and Li Yuan. Masked autoencoders for point cloud self-supervised learning. In *ECCV*, 2022. 2

[27] Xumin Yu, Lulu Tang, Yongming Rao, Tiejun Huang, Jie Zhou, and Jiwen Lu. Point-BERT: Pre-training 3D point cloud transformers with masked point modeling. In *CVPR*, 2022. 2

[28] Ji Hou, Benjamin Graham, Matthias Nießner, and Saining Xie. Exploring data-efficient 3d scene understanding with contrastive scene contexts. In *CVPR*, 2021. 2

[29] Xiaoyang Wu, Xin Wen, Xihui Liu, and Hengshuang Zhao. Masked scene contrast: A scalable framework for unsupervised 3d representation learning. In *CVPR*, 2023. 2

[30] Haoyi Zhu, Honghui Yang, Xiaoyang Wu, Di Huang, Sha Zhang, Xianglong He, Tong He, Hengshuang Zhao, Chunhua Shen, Yu Qiao, et al. Ponderv2: Pave the way for 3d foundataion model with a universal pre-training paradigm. *arXiv:2310.08586*, 2023. 2

[31] Zhuoxiao Chen, Yadan Luo, Zi Huang, Zheng Wang, and Mahsa Baktashmotlagh. Revisiting domain-adaptive 3d object detection by reliable, diverse and class-balanced pseudo-labeling. In *ICCV*, 2023. 2

[32] Jiakang Yuan, Bo Zhang, Xiangchao Yan, Tao Chen, Botian Shi, Yikang Li, and Yu Qiao. Bi3d: Bi-domain active learning for cross-domain 3d object detection. In *CVPR*, 2023. 2

[33] George Eskandar, Chongzhe Zhang, Abhishek Kaushik, Karim Guirguis, Mohamed Sayed, and Bin Yang. An empirical study of the generalization ability of lidar 3d object detectors to unseen domains. *arXiv:2402.17562*, 2024. 2

[34] 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. *NeurIPS*, 2022. 2

[35] Ziyu Li, Jingming Guo, Tongtong Cao, Liu Bingbing, and Wankou Yang. Gpa-3d: Geometry-aware prototype alignment for unsupervised domain adaptive 3d object detection from point clouds. In *ICCV*, 2023. 2

[36] Yan Wang, Xiangyu Chen, Yurong You, Li Erran Li, Bharath Hariharan, Mark E. Campbell, Kilian Q. Weinberger, and Wei-Lun Chao. Train in germany, test in the USA: making 3d object detectors generalize. In *CVPR*, 2020. 2, 8

[37] Yi Wei, Zibu Wei, Yongming Rao, Jiaxin Li, Jie Zhou, and Jiwen Lu. Lidar distillation: Bridging the beam-induced domain gap for 3d object detection. In *ECCV*, volume 13699, 2022. 3

[38] Qiangeng Xu, Yin Zhou, Weiyue Wang, Charles R Qi, and Dragomir Anguelov. Spg: Unsupervised domain adaptation for 3d object detection via semantic point generation. In *ICCV*, 2021. 3

[39] Zeng Yihan, Chunwei Wang, Yunbo Wang, Hang Xu, Chaoqiang Ye, Zhen Yang, and Chao Ma. Learning transferable features for point cloud detection via 3d contrastive co-training. *NeurIPS*, 2021. 3

[40] Alexey Dosovitskiy, German Ros, Felipe Codevilla, Antonio Lopez, and Vladlen Koltun. Carla: An open urban driving simulator. In *CoRL*, 2017. 3, 5, 7

[41] Bernhard Wymann, Eric Espié, Christophe Guionneau, Christos Dimitrakakis, Rémi Coulom, and Andrew Sumner. Torcs, the open racing car simulator. *Software available at http://www.torcs.org*, 2000. 3

[42] Shital Shah, Debadeepta Dey, Chris Lovett, and Ashish Kapoor. Airsim: High-fidelity visual and physical simulation for autonomous vehicles. In *Field and Service Robotics*, 2018. 3

[43] Jean-Emmanuel Deschaud. Kitti-carla: a kitti-like dataset generated by carla simulator. *arXiv:2109.00892*, 2021. 3

[44] Nathan Koenig and Andrew Howard. Design and use paradigms for gazebo, an open-source multi-robot simulator. In *IROS*, 2004. 3

[45] 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 *ICMR*, 2018. 3

[46] 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. 3

[47] Chenqi Li, Yuan Ren, and Bingbing Liu. Pcgen: Point cloud generator for lidar simulation. In *ICRA*, 2023. 3, 6

[48] Ze Yang, Yun Chen, Jingkang Wang, Sivabalan Manivasagam, Wei-Chiu Ma, Anqi Joyce Yang, and Raquel Urtasun. Unisim: A neural closed-loop sensor simulator. In *CVPR*, 2023. 3

[49] Tang Tao, Longfei Gao, Guangrun Wang, Peng Chen, Dayang Hao, Xiaodan Liang, Mathieu Salzmann, and Kaicheng Yu. Lidar-nerf: Novel lidar view synthesis via neural radiance fields. *arXiv:2304.10406*, 2023. 3

[50] Yan Yan, Yuxing Mao, and Bo Li. SECOND: sparsely embedded convolutional detection. *Sensors*, 2018. 3, 8, 9

[51] Bo Zhang, Xinyu Cai, Jiakang Yuan, Donglin Yang, Jianfei Guo, Xiangchao Yan, Renqiu Xia, Botian Shi, Min Dou, Tao Chen, Si Liu, Junchi Yan, and Yu Qiao. Resimad: Zero-shot 3d domain transfer for autonomous driving with source reconstruction and target simulation. In *ICLR*, 2024. 5, 7, 8, 14

[52] Jeong Joon Park, Peter Florence, Julian Straub, Richard A. Newcombe, and Steven Lovegrove. Deepsdf: Learning continuous signed distance functions for shape representation. In *CVPR*, 2019. 5, 14

[53] Yibo Liu, Kelly Zhu, Guile Wu, Yuan Ren, Bingbing Liu, Yang Liu, and Jinjun Shan. Mv-deepsdf: Implicit modeling with multi-sweep point clouds for 3d vehicle reconstruction in autonomous driving. In *ICCV*, 2023. 5

[54] Angel X. Chang, Thomas A. Funkhouser, Leonidas J. Guibas, Pat Hanrahan, Qi-Xing 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. *arXiv:1512.03012*, 2015. 5, 14

[55] Peng Wang, Lingjie Liu, Yuan Liu, Christian Theobalt, Taku Komura, and Wenping Wang. Neus: Learning neural implicit surfaces by volume rendering for multi-view reconstruction. In *NeurIPS*, 2021. 5, 15

[56] Francis Williams, Zan Gojcic, Sameh Khamis, Denis Zorin, Joan Bruna, Sanja Fidler, and Or Litany. Neural fields as learnable kernels for 3d reconstruction. In *CVPR*, 2022. 5

[57] Jiahui Huang, Zan Gojcic, Matan Atzmon, Or Litany, Sanja Fidler, and Francis Williams. Neural kernel surface reconstruction. In *CVPR*, 2023. 5, 15

[58] Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. NeRF: Representing scenes as neural radiance fields for view synthesis. In *ECCV*, 2020. 6

[59] I. Vizzo, X. Chen, N. Chebrolu, J. Behley, and C. Stachniss. Poisson Surface Reconstruction for LiDAR Odometry and Mapping. In *ICRA*, 2021. 7

[60] Vlas Zyrianov, Xiyue Zhu, and Shenlong Wang. Learning to generate realistic lidar point clouds. In *ECCV*, 2022. 7

[61] Yuwen Xiong, Wei-Chiu Ma, Jingkang Wang, and Raquel Urtasun. Learning compact representations for lidar completion and generation. In *CVPR*, 2023. 7

[62] 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 *CVPR*, 2020. 8, 9

[63] Zhiqin Chen and Hao Zhang. Learning implicit fields for generative shape modeling. In *CVPR*, 2019. 14

# LiT: Unifying LiDAR "Languages" with LiDAR Translator Supplementary Materials

## Contents



## A.1 Foreground modeling details

**Tracking and fusion.** Reconstructing foreground objects directly from each LiDAR frame is suboptimal as it does not take into account the multiple viewing angles of the object as the LiDAR sensor or the object moves. To address this issue, we fuse the point clouds from multiple LiDAR frames to reconstruct the foreground objects. Typically, tracking information is provided as unique object identifiers in the LiDAR datasets [9, 10, 11], where vehicles are tracked across multiple frames using these identifiers. For each identified vehicle, its point clouds from different frames are aligned based on their bounding box information and then fused.

**Mesh reconstruction.** With multiple LiDAR frames, the fused point clouds are incomplete and noisy due to the sparsity and scanning gaps inherent in LiDAR data. To faithfully reconstruct the foreground objects while addressing the noise and sparsity issues, we need to introduce prior knowledge of vehicle geometries. Inspired by the success of neural implicit surface reconstruction for object shape modeling [63, 52], we first train an SDF-based model with the ShapeNetV2 [54] vehicle category and then apply this model to reconstruct the foreground object meshes. In essence, given a latent code, the model maps 3D points to a signed distance field (SDF) representing the mesh surface, where the key here is to solve for the latent code given multi-view point cloud observations.

Given a set of points from LiDAR scans, $\{\boldsymbol{x}_i\}_{i=1}^{N}$, our goal is to find a latent code, $\boldsymbol{z}$, that best represents the underlying surface that these points belong to. This optimization process is guided by minimizing the Signed Distance Function (SDF) loss, defined as:

$$\mathcal{L}_{\text{SDF}} = \frac{1}{N}\sum_{i=1}^{N}\left(\left\|f_\theta\left(\boldsymbol{z}, \boldsymbol{x}_i\right) - 0\right\|_1\right) + \lambda\|\boldsymbol{z}\|_2^2,$$

where $f_\theta\left(\boldsymbol{z}, \boldsymbol{x}_i\right)$ denotes the predicted SDF value for point $\boldsymbol{x}_i$ given the latent code $\boldsymbol{z}$, and the target SDF value for points on the object surface is set to zero. The term $\lambda\|\boldsymbol{z}\|_2^2$ adds an L2 regularization on the latent code to encourage generalization by penalizing its magnitude, with $\lambda$ being the regularization coefficient.

Notably, as our mesh is reconstructed from point clouds, it is not limited to using human-created vehicle 3D assets, which is a key difference from ReSimAD [51]. This optimization process effectively adapts the generic vehicle geometry learned from ShapeNet to fit the specific geometry of the vehicle represented in the sparse and noisy LiDAR point cloud. By iteratively refining the latent code, we can generate a mesh that accurately captures the detailed geometry of the vehicle, even in the presence of data sparsity and noise inherent to LiDAR scans. We show some example visualizations of foreground modeling in Fig. 8.

The hyperparameters for scene modeling are shown in Table 7. For background modeling, while most parameters are consistent across both Waymo and nuScenes datasets, a key difference is their frame rates. Specifically, every other frame is skipped in the Waymo dataset, which is attributed to its denser point clouds and a higher LiDAR frame rate of 10Hz, in contrast to the 2Hz frame rate of nuScenes. Consequently, we sample the Waymo dataset at 5Hz to ensure a comparable point cloud density and maintain reconstruction quality across datasets.

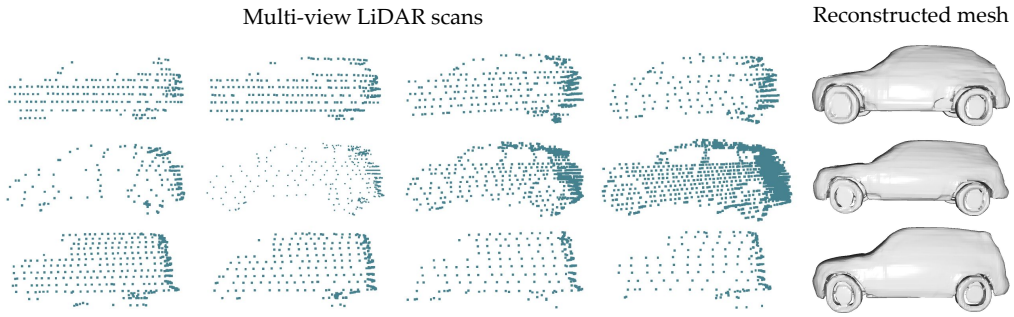


Figure 8: **Foreground modeling samples.** We show some examples of foreground modeling with LiT. The left columns show the original LiDAR point clouds collected from multiple LiDAR frames. The rightmost column shows the reconstructed mesh from the multi-view LiDAR inputs. The reconstructed mesh will then be used by LiT to perform target-domain LiDAR ray casting.

## A.2 Background modeling details

Next, we reconstruct the background scene mesh with point clouds from multiple LiDAR frames. For each LiDAR frame point cloud, foreground objects are removed based on the bounding box information. Then, point clouds from multiple frames are transformed and fused in world coordinates by the ego vehicle's pose and LiDAR-to-ego pose.

To reconstruct the background scene mesh from the fused point cloud, we follow neural kernel field [57], where the reconstructed shape is modeled as the zero level set of a 3D implicit field $f_\theta : \mathbb{R}^3 \to \mathbb{R}$ defined as a hierarchical neural kernel field. This field combines positive definite kernels, weighted and conditioned on the inputs, and centered at the midpoints of voxels within the predicted hierarchy:

$$f_\theta(\mathbf{x}|\mathcal{X}_{in}, \mathcal{N}_{in}) = \sum_{i,l} \alpha_i^{(l)} K_\theta^{(l)}(\mathbf{x}, \mathbf{x}_i^{(l)}|\mathcal{X}_{in}, \mathcal{N}_{in}),$$

where $\alpha_i^{(l)}$ are scalar coefficients at the $i^{th}$ voxel at level $l$ in the hierarchy. Then, the kernel for the $l^{th}$ level, $K_\theta^{(l)}$, is defined as:

$$K_\theta^{(l)}(\mathbf{x}, \mathbf{x}') = \langle \phi_\theta^{(l)}(\mathbf{x}|\mathcal{X}_{\text{in}}, \mathcal{N}_{\text{in}}), \phi_\theta^{(l)}(\mathbf{x}'|\mathcal{X}_{\text{in}}, \mathcal{N}_{\text{in}}) \rangle \cdot K_{\text{b}}^{(l)}(\mathbf{x}, \mathbf{x}').$$

Given the voxel hierarchy and predicted normals, the implicit surface is computed by minimizing a specific loss function to find optimal coefficients. This process aims to align the neural kernel field gradient with the normals at voxel centers and to approximate zero at all input points. We employ the pre-trained kitchen-and-sink model which is able to generalize to diverse datasets. For a fair comparison, the model has not been trained on the LiDAR datasets that we use in this work. Compared with ReSimAD with their NeuS [55]-based background modeling, our approach is more efficient as it does not require custom training for each scene.

The hyperparameters for scene modeling are shown in Table 7. Additional visualizations of the background modeling are provided in Fig. 9. For foreground modeling, we utilize a fixed number of steps and a predefined learning rate to reconstruct the foreground objects. We use the same parameters for both Waymo and nuScenes datasets.

## A.3 LiDAR statistical modeling visualization

As discussed in Sec. 4.2, we conduct a statistical modeling of LiDAR rays to model the distribution of the vertical angles. We identify the peak angles where the majority of the LiDAR rays are concentrated and generate rays based on these identified peak angles. This ensures that our simulated LiDAR data is representative of real-world LiDAR characteristics. We illustrate this effect visually in Fig. 10, while the LiDAR statistics of each dataset are presented in Fig. 2.

Table 7: **Parameters for scene modeling.** The left table shows the parameters for background modeling. The right table shows the parameters for foreground modeling.

| Background parameters | Waymo | nuScenes |
|---|---|---|
| Skip every N frames | 2 | 1 |
| LiDAR frame rate | 5Hz | 2Hz |
| Enabled LiDARs | [TOP] | [TOP] |
| Voxel size | 0.25 | 0.25 |
| Normal estimation kNN | 64 | 64 |
| Normal estimation drop angle | 85° | 85° |
| Solver max iteration | 2,000 | 2,000 |
| Solver convergence tolerance | 1e-5 | 1e-5 |

| Foreground parameters | Value |
|---|---|
| Latent code optimizer iterations | 500 |
| Optimizer learning rate | 5e-5 |
| Number of samples per iteration | 8000 |
| Meshing voxel resolution | 128 |
| Maximum batch size | $32^3$ |
| Latent code clamp distance | 0.1 |
| Statistical outlier kNN | 20 |
| Statistical outlier std ratio | 1.0 |

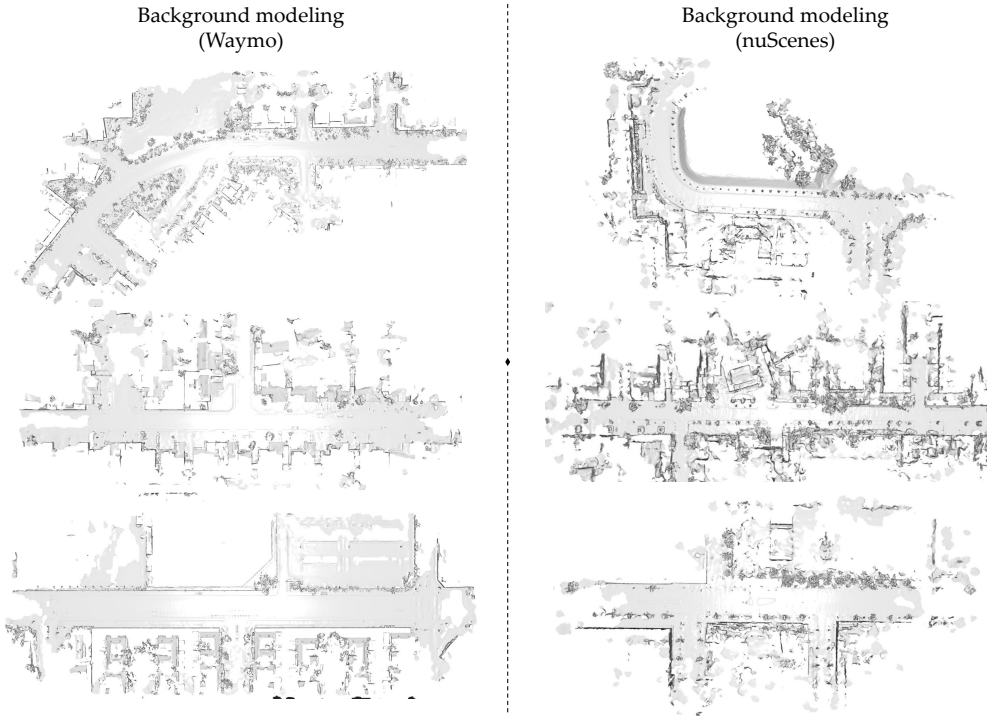


Figure 9: **Background modeling samples.** We provide additional visualization samples of background modeling for Waymo and nuScenes.

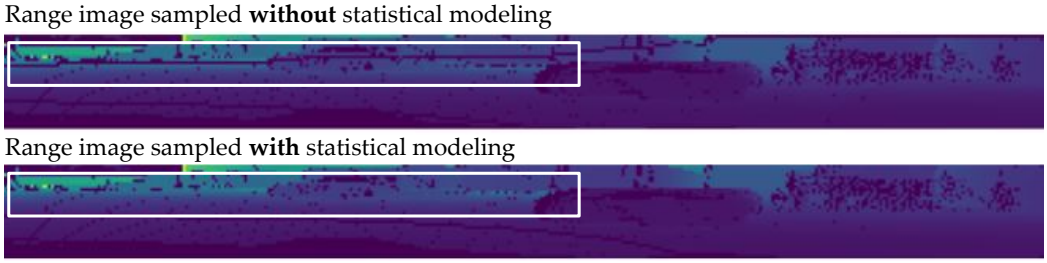


Figure 10: **Effects of LiDAR statistical modeling visualized with range image.** We illustrate the effect of statistical modeling of LiDAR ray angles with a scene from nuScenes. The top row shows the 2D range image sampled without statistical modeling, and the bottom row shows the 2D range image sampled using statistical modeling. The bottom row shows fewer artifacts (e.g. the horizontal gap) as the sampled rays are more concentrated around the peak angles.

## A.4 Evaluation of scene and LiDAR modeling

We perform evaluation on the quality of the scene and LiDAR modeling as an integrated system. To do this, we perform a self-to-self translation task on the source domain, where we use LiT to reconstruct the scene and perform LiDAR ray casting using the same dataset's LiDAR parameters. We then calculate the Chamfer distance between the original point cloud $P_{\text{gt}}$ and the ray casted point cloud $P_{\text{sim}}$:

$$\text{CD}(P_{\text{sim}}, P_{\text{gt}}) = \frac{1}{|P_{\text{sim}}|} \sum_{\mathbf{x} \in P_{\text{sim}}} \min_{\mathbf{y} \in P_{\text{gt}}} \|\mathbf{x} - \mathbf{y}\|_2^2 + \frac{1}{|P_{\text{gt}}|} \sum_{\mathbf{y} \in P_{\text{gt}}} \min_{\mathbf{x} \in P_{\text{sim}}} \|\mathbf{y} - \mathbf{x}\|_2^2.$$

The distributions of the Chamfer distances for the Waymo and nuScenes datasets are shown in Fig. 11.

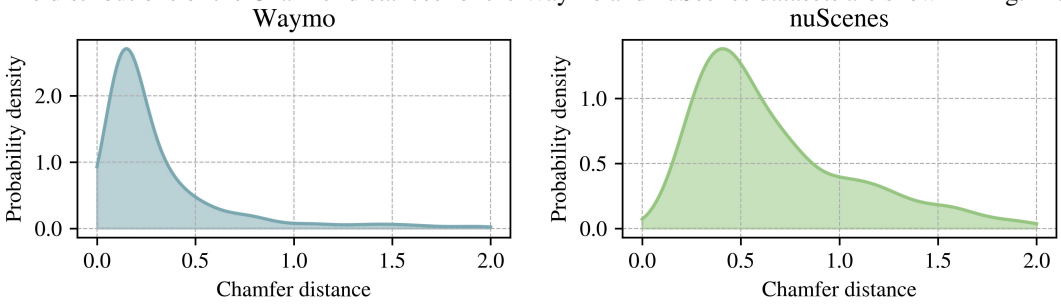


Figure 11: **Chamfer distance distributions.** We evaluate the quality of the scene and LiDAR modeling. For a given dataset, we use LiT to reconstruct the scene and perform a self-to-self translation of the scene. We then measure the Chamfer distance between the original and the translated point cloud. The distributions of the Chamfer distances for the Waymo and nuScenes datasets are shown. In general, Waymo has lower Chamfer distances compared to nuScenes, which is attributed to the denser point cloud from the higher LiDAR resolutions and sampling rate in Waymo.

## A.5 Training details

We provide hyperparameters used for training in Table 8.

Table 8: **Training hyperparameters for domain adaptation tasks.** This summary presents the hyperparameters for training detection models under single-source domain unification settings. The table includes configurations for Second-IOU and PV-RCNN models.

| | Waymo → KITTI | | Waymo → nuScenes | | nuScenes → KITTI | |
|---|---|---|---|---|---|---|
| | Second-IOU | PV-RCNN | Second-IOU | PV-RCNN | Second-IOU | PV-RCNN |
| Optimizer | Adam | Adam | Adam | Adam | Adam | Adam |
| Scheduler | One-Cycle | One-cycle | One-csycle | One-Cycle | One-Cycle | One-Cycle |
| Learning rate | 1e-4 | 1e-4 | 1e-4 | 1e-4 | 1e-4 | 1e-4 |
| Momentum | 0.9 | 0.9 | 0.9 | 0.9 | 0.9 | 0.9 |
| Weight decay | 1e-2 | 1e-3 | 1e-2 | 1e-3 | 1e-2 | 1e-3 |
| Batch size | 32 | 16 | 32 | 16 | 32 | 16 |
| Epochs (ft.) | 10 | 10 | 10 | 10 | 10 | 10 |

## 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: We confirm that the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope.

   Guidelines:

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

2. **Limitations**

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

   Answer: [Yes]

   Justification: We discuss the limitations of our work in the conclusion section of the paper.

   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: We do not include theoretical results in our paper.

Guidelines:

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

4. **Experimental Result Reproducibility**

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

   Answer: [Yes]

   Justification: We describe the data processing, experimental setup, and evaluation metrics in detail in the main paper and supplemental material.

   Guidelines:

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

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

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

Answer: [Yes]

Justification: We provide access to the code, with instructions on data preparation, training and evaluation.

Guidelines:

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

6. **Experimental Setting/Details**

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

   Answer: [Yes]

   Justification: We provide detailed information on the dataset, hyperparameters, and evaluation metrics in the main paper and supplemental material.

   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: [NA]

   Justification: Our evaluation and comparison are based on standard metrics as used in prior works in the domain. We have not included statistical significance tests in the paper.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)

- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments Compute Resources**

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

   Answer: [Yes]

   Justification: We provide hardware details and runtime statistics for our pipeline.

   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: We reviewed the NeurIPS Code of Ethics and confirm that our research conforms to the guidelines.

   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: We discuss the societal impacts of our work in the conclusion section of the paper.

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.

- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

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

    Answer: [NA]

    Justification: The paper does not pose 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: We cite and respect the licenses of all assets used in the paper.

    Guidelines:

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

- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New Assets**

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

    Answer: [NA]

    Justification: We do not introduce new assets in the paper.

    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 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.