2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
October 14-18, 2024. Abu Dhabi, UAE

Enhancing LiDAR Scene Upsampling with Instance-aware Feature-embedding and Attention Mechanism

Wei-Jen Wang[1], You-Sheng Do[1], Wen-Chieh Lin[1] and Chieh-Chih Wang[2]

***Abstract*— Scanning LiDAR is one of the widely used sensors in autonomous vehicles; however, the inherent sparsity of LiDAR point clouds often affects its performance. To address this issue, upsampling methods could be employed to enhance low-resolution LiDAR data. Although there have been methods on upsampling of single-object point clouds recently in computer vision, they tend to generate a considerable amount of artifacts when dealing with real-world LiDAR scenes consisting of multiple objects. In this paper, we propose a solution to tackle this problem by introducing an instance embedding auxiliary task and a context attention module. With our auxiliary learning architecture, the network can learn features that benefit both the primary upsampling task and the auxiliary instance embedding task. This training design enables the point generation process to be carried out separately and significantly reduces artifacts of the upsampling results on the SemanticKITTI dataset, particularly in areas surrounding instances. By leveraging these techniques to improve the model's understanding of the relationship between objects and the background in LiDAR scenes, we achieve an overall 4% to 10% improvement in whole-scene upsampling.**

I. Introduction

In recent years, LiDAR has become a popular sensor in autonomous vehicles for surrounding detection, recognition, and prediction. Among the different types of LiDARs, scanning LiDAR is commonly used in a number of public datasets such as SemanticKITTI [1], but it suffers from sparse point clouds. To address this issue, one possible solution is to employ upsampling methods to increase the density of the LiDAR point cloud. Some recent studies on point cloud upsampling have focused on point clouds generated from single objects (e.g., ShapeNet [2] and ModelNet40 [3]), where pairs of point clouds, sampled from CAD objects with high- and low-resolution versions, are provided for training and testing. These datasets typically feature a single object in each scene, and the data pairs are uniformly sampled from object meshes. Consequently, these upsampling methods may not be suitable for LiDAR scenes captured in the real world.

LiDAR scenes captured from the real world are much more complex, encompassing various objects and background elements spread throughout the entire scene. The intricate nature of LiDAR point clouds presents challenges when state-of-the-art upsampling methods [4]–[6] are directly applied to LiDAR point clouds. For example, Neural Points (NPT) [4] in Figure 1 generate numerous artifacts, especially in the regions between objects. One reason for this is that these methods struggle to distinguish different objects and often mistake multiple objects for a single entity.

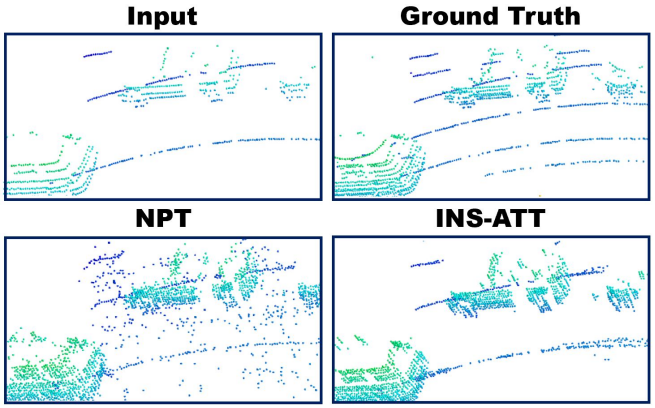

Fig. 1: Top row shows input LiDAR data downsampled from a LiDAR scene (treated as the ground truth of upsampling). Compared to Neural Points (NPT) [4], our method, INS-ATT, successfully reduces the artifacts between objects in the upsampled results.

Another challenge lies in the sparsity of LiDAR point clouds, where objects far from the LiDAR center are often represented by only a few points. Previous method [4]–[6] frequently produce inaccurate contours in such cases due to limitations in nearest neighbor search within their feature extractors [7]–[9], which may mistakenly mix points from different objects in sparse areas. Consequently, without a global perspective, the association of points often lacks meaningful connections and tends to overlook the overall shape distribution.

In this paper, we introduce two enhancement strategies to address the challenges mentioned above. First, to mitigate the issue that arises from the ambiguous belonging of different objects, **our proposed method adopts an auxiliary learning framework [10]–[12] to integrate instance priors into the representation learning of the network**. Within this framework, an additional auxiliary task involving instance embedding is concurrently learned alongside the primary upsampling task. By grouping instances in the feature space, this auxiliary task forces the feature-extraction modules to be aware of the instance belonging of each point. It is noteworthy that the incorporation of instance embedding is only utilized during the training process to enrich the model's understanding and segmentation capabilities. However, our method is designed to be fully operational in LiDAR scenes without instance information. This ensures that, once trained,

[1]Wei-Jen Wang, You-Sheng Do, and Wen-Chieh Lin are with the College of Computer Science, National Yang Ming Chiao Tung University, Hsinchu, Taiwan. willy8898.cs09g@nctu.edu.tw; aaaa95067.cs12@nycu.edu.tw; wclin@cs.nctu.edu.tw

[2]Chieh-Chih Wang is with the College of Electrical and Computer Engineering, National Yang Ming Chiao Tung University, and with the Mechanical and Mechatronics Systems Research Laboratories, Industrial Technology Research Institute, Hsinchu, Taiwan. bobwang@ieee.org

the model can be applied in real-world scenarios where instance labels are not readily available, highlighting our approach's flexibility and broad applicability.

Second, to address the limitation of a small receptive field, **our proposed method integrates a context attention module, which employs a self-attention mechanism to facilitate interaction among all points within the subscene input**. As a result, each local region gains access to a richer context, allowing the network to achieve a more comprehensive global perspective. Ultimately, the output of the context module refines the features extracted from the local feature extractor, reducing the weight of irrelevant points. This allows our upsampling phase to ignore irrelevant points and generate more accurate shapes in sparse areas. While the self-attention mechanism has been used in semantic segmentation and object detection in LiDAR data, our work represents the first endeavor to apply self-attention to LiDAR scene upsampling. This novel application of self-attention offers a promising avenue for improving LiDAR data upsampling in complex, multi-object scenarios.

We conducted comprehensive evaluations using the SemanticKITTI dataset. In the whole scene, our method outperforms previous methods [4]–[6], resulting in an improvement in voxel IoU ranging from 4% to 10%. Furthermore, to validate the reduction of artifacts around instances, we assessed the voxel IoU for sub-scenes encompassing each instance. The results for the three primary instance classes within SemanticKITTI demonstrated remarkable improvements. This significant advancement underscores the comprehensive impact of our method in enhancing the overall understanding of the scene.

The main contributions of this work include: (1) We propose a method to address the 3D upsampling problems of real-world LiDAR scenes with complicated and multiple objects, which have been less addressed. (2) We propose to incorporate instance embedding auxiliary task into the training process of an upsampling network, which is able to learn instance-aware representations. (3) We propose to add the context attention module to properly determine the attention range and mitigate the wrong shape generation in the sparse areas.

II. Related Work

A. Object upsampling

Most learning-based point cloud upsampling methods primarily focus on datasets derived from artificial object datasets such as ShapeNet [2] and ModelNet40 [3]. These datasets typically feature scenes with a single object, and the point cloud shapes are often spatially uniform. PUNet [13] can be considered a pioneering work in this field. They utilized PointNet [7] to generate multi-level features and expanded the point set using multi-branch MLPs. MPU [6] divided the upsampling process into multiple two-time upsampling steps. PU-GAN [14] introduced an additional discriminator network and formulated the upsampling task as an adversarial problem. PU-GCN [5] proposed the NodeShuffle module to expand the point set and Inception DenseGCN to extract features from different receptive fields. To achieve an arbitrary upsampling ratio, NPT [4] constructed a neural field to represent the continuous surface from the point set and formulated the upsampling as resampling on this surface.

B. Scene upsampling

When working with LiDAR scenes like SemanticKITTI [1], the point clouds exhibit distinct characteristics that differ markedly from those of artificial objects. These scenes involve complex scenarios with multiple objects and classes, along with non-uniform and sparser spatial distributions. Most LiDAR upsampling methods [15]–[18] convert the entire LiDAR scene into range image representations, using image super-resolution techniques to infer depth. However, 2D networks that process range images can struggle with neighboring pixels that have significantly different depths, leading to the generation of unrelated noise with interpolated depth.

However, the exploration of upsampling directly on raw LiDAR scenes remains relatively limited. While these approaches generally follow a common pipeline for point cloud upsampling, they stand out by addressing the unique characteristics of LiDAR point clouds. For instance, LiUpNet [19] introduces a neighbor aggregation block to capture features through multi-scale information and incorporates a consistency loss to enhance the model's robustness to varying point cloud densities. In [20], the limitations of traditional upsampling methods for reconstructing LiDAR patterns are discussed. To overcome these limitations, the concept of "Sliced Wasserstein Distance" [21]–[23] is introduced for upsampling LiDAR patches.

To the best of our knowledge, no one has considered instance relationships within the LiDAR scene in conjunction with the upsampling task on raw point clouds. Without awareness of different instances, the upsampling process may treat two different instances as one, leading to artifacts generated around closely spaced instances. In our method, the instance auxiliary task helps the model implicitly encode instance associations into the final feature. This enables the separate treatment of different instances, effectively reducing the likelihood of generating artifacts.

III. Method

A simple approach to upsample a LiDAR scene with multiple objects may be to segment the point cloud of each object and then upsampling them independently. However, this often fails because conventional object segmentation methods struggle with sparse LiDAR data, which is where upsampling is typically needed. Therefore, our solution integrates instance segmentation with upsampling during training to improve outcomes. Based on Neural Points (NPT) [4], we propose adding two important components: the context attention module and the auxiliary learning with instance embedding. Figure 2 is an overview of the proposed method.

A. Background on Neural Points (NPT)

Similar to recent methods on point cloud upsampling [4]–[6], [13], [14], NPT also follows a three-step computational

pipeline. First, a LiDAR scene to be upsampled is divided into L sparse patches. These patches form the sparse set denoted by $S = \{s_j\}_{j=1}^{L}$. Each patch containing N_j points, i.e., $s_j = \{p_n \in \mathbb{R}^3\}_{n=1}^{N_j}$, is treated as an input unit. Second, each patch s_j is fed into the upsampling network, generating a dense patch $d_j = \{q_n \in \mathbb{R}^3\}_{n=1}^{rN_j}$, where $r \geq 1$ is the upsampling ratio. Third, the dense patch set $D = \{d_j\}_{j=1}^{L}$ is concatenated and further sampled to the required point number using the farthest point sampling [24] to form the final upsampled scene.

NPT consists of two modules: (1) The local feature aggregation generates local features by concatenating the outputs from DGCNN [9] and the position encoding module; (2) The upsampler estimates a potential neural field based on the local features. This neural field can be considered as an implicit function that establishes a bijective map between a 2D parametric domain and a 3D continuous surface. Since upsampling is framed as a point-sampling problem on the estimated continuous surface, the upsampling ratio can be flexible, eliminating the need for frequent retraining.

We made three modifications to NPT so that it can handle LiDAR point cloud datasets. First, NPT utilizes the normal at each point to compute the loss function. However, for LiDAR point cloud, estimated normals are often noisy and challenging to be used directly in model training. Therefore, the loss function is replaced by the Chamfer distance. Second, in the original NPT, after the upsampler outputs the 3D query points using the local neural field, these 3D query points are further projected to the global neural field through the distance-weighted sum of nearest patch centers. This process, which is called Neural Field Integration, is appropriate for single-object upsampling because the points should all lie on the same object surface. However, a LiDAR scene often contains many objects, so the assumption of a single object is not reasonable. Consequently, we do not adopt the Neural Field Integration process.

B. Context attention module

In NPT [4], the network was specifically designed for upsampling the surface of a single object. While such surfaces may not always be uniformly sampled, they generally exhibit relatively consistent density and are free from occlusion. In these cases, a local feature extractor like DGCNN [9] is sufficient to achieve good performance. However, in regions farther from the LiDAR sensor, object structures are often incomplete, and only a few points are captured. Since DGCNN [9] relies on K-nearest neighbors (KNN) to find a fixed number of neighbors, points in these sparse regions might be linked to distant, unrelated neighbors. As shown in Figure 3, without a mechanism to reweight the importance of the referenced points, the upsampling process may be skewed by irrelevant points, leading to incorrect upsampling shapes.

We propose a context attention module using a self-attention mechanism to address this issue. The module draws inspiration from the local transformer [25]. Our context module comprises multi-head self-attention followed by MLPs. The multi-head attention block considers all the points in the patch and utilizes the point-wise local features from the local feature aggregation output. Unlike the original design in [25], we omitted the maxpooling layer after the multi-head self-attention blocks. The maxpooling layer was originally designed to aggregate high-level features and assist in proposing potential object centroids for 3D object detection. However, in the upsampling task, attention to detail is more critical, so we directly utilize the point-wise attention values from the multi-head attention module.

Algorithm 1 Self-attention in context attention module

Input: The point-wise local feature $\mathbf{F} = [\mathbf{f}_1, \mathbf{f}_2] \in \mathbb{R}^{N_j \times C}$ from the local feature aggregation block , where $[...]$ is the concatenation operation; $\mathbf{f}_1 \in \mathbb{R}^{N_j \times C_1}$ is the point-wise feature from DGCNN and $\mathbf{f}_2 \in \mathbb{R}^{N_j \times C_2}$ is the position encoding vector;

Output: The adjustment vector $\mathbf{A}' \in \mathbb{R}^{N_j \times C}$;

$\mathbf{Q} = \mathbf{F}\mathbf{W}_Q$;
$\mathbf{K} = \mathbf{F}\mathbf{W}_K$;
$\mathbf{V} = \mathbf{F}\mathbf{W}_V$;
for $i = 1$ to t **do**
    $\mathbf{h}_i = Attention(\mathbf{Q}\mathbf{H}_Q^i, \mathbf{K}\mathbf{H}_K^i, \mathbf{V}\mathbf{H}_V^i)$;
end for
$\mathbf{A} = [\mathbf{h}_1, \ ... \ \mathbf{h}_t] \cdot \mathbf{W}_o$;
$\mathbf{A}' = MLPs(\mathbf{A})$;
return $\mathbf{A}'$;

Specifically, our multi-head self-attention can be expressed as Algorithm 1. Given the local feature $\mathbf{F} \in \mathbb{R}^{N_j \times C}$, extracted by the local feature aggregation block from the sparse patch s_j containing N_j points, our goal is to obtain the adjustment vector $\mathbf{A}' \in \mathbb{R}^{N_j \times C}$ for $\mathbf{F}$. Here, $\mathbf{F}$ is the concatenation of DGCNN's feature $\mathbf{f}_1 \in \mathbb{R}^{N_j \times C_1}$ and the position encoding vector $\mathbf{f}_2 \in \mathbb{R}^{N_j \times C_2}$. $C = C_1 + C_2$.

Since the input is solely derived from the same local feature $\mathbf{F}$, it represents a special case of attention known as self-attention. We apply linear projections to $\mathbf{F}$ using the corresponding projection matrices $\mathbf{W}_Q$, $\mathbf{W}_K$, $\mathbf{W}_V \in \mathbb{R}^{C \times D}$ to obtain $\mathbf{Q}$, $\mathbf{K}$, and $\mathbf{V} \in \mathbb{R}^{N_j \times D}$. Essentially, $\mathbf{Q}$ is the query attribute of the point feature $\mathbf{F}$, while $\mathbf{K}$ represents the local region characteristic around points in patch s_j. In the context of plain self-attention, $\mathbf{Q}$ and $\mathbf{K}$ are used to calculate similarity through inner product, achieving the "re-weighting" of points. Subsequently, the feature similarity serves as the weight and is multiplied by the corresponding refinement offset in $\mathbf{V}$ to gather the comprehensive attention information. The attention operation is defined by

$$Attention(\mathbf{q}, \mathbf{k}, \mathbf{v}) = softmax(\mathbf{q}\mathbf{k}^T)\mathbf{v}. \tag{1}$$

where $\mathbf{q}$, $\mathbf{k}$, and $\mathbf{v}$ are query, key, and value matrix, respectively.

To model different types of attention, multiple heads are used for $\mathbf{Q}$, $\mathbf{K}$, and $\mathbf{V}$. For each head i, these three matrices undergo further projection using the corresponding head projection matrices $\mathbf{H}_Q^i$, $\mathbf{H}_K^i$, $\mathbf{H}_V^i \in \mathbb{R}^{D \times E}$. Following Equation 1, we apply the attention operation to get the output $\mathbf{h}_i \in \mathbb{R}^{N_j \times E}$. After calculating the attention values

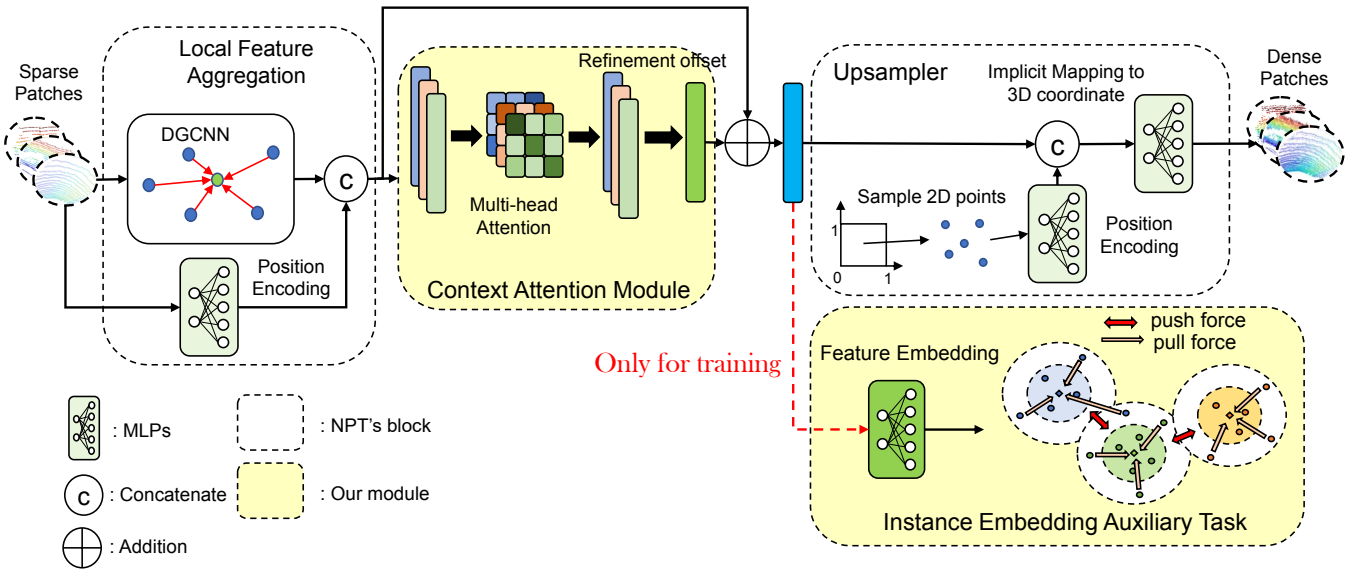

Fig. 2: Our upsampling network architecture overview. The baseline network consists of the local feature aggregation and upsampler. Building on the baseline network, we incorporate two additional components (outlined in yellow boxes): the context attention module and the instance embedding auxiliary task.

for each head, the multi-head attention module concatenates all outputs and aggregates them using the linear projection matrix $\mathbf{W}_o \in \mathbb{R}^{tE \times D}$. Finally, MLP layers are applied to the aggregated attention value to construct the adjustment vector $\mathbf{A}'$.

Through the self-attention operation, the context attention module enhances the correlation among all the points in the patch. For neighbor points that are distant from the query point, the module learns to reduce their weights and puts more emphasis on the nearer points. The final adjustment vector $\mathbf{A}'$ is added to the local feature. With this adjustment, the model can reduce the influences of unrelated points and ultimately disregard these points in the subsequent upsampling process.

C. Auxiliary learning with instance embedding

From machine learning perspective, auxiliary learning (AL) is a means of achieving inductive transfer, where inductive bias is incorporated into the feature learning process. This inductive bias serves as a form of regularization, guiding the model to acquire relevant knowledge that can benefit the main task. In practice, AL achieves this goal by introducing additional objectives for the auxiliary task.

Our auxiliary learning framework, as illustrated in Figure 2, utilizes instance embedding as the auxiliary branch exclusively during the training phase. The main upsampling task and the auxiliary task share the same features from the context attention module. By sharing features between the tasks, AL allows the network to perform the upsampling task while being aware of instance relationships, enhancing the network's ability to distinguish between objects.

Instance embedding in our framework was inspired by [26]. The idea behind it is that the model will learn a differentiable function to map each 3D point to a point in a high-dimensional feature space, referred to as the *embedding point*. Ideally, the embedding points with the same instance label should be close to each other in the embedding space, while the points with different instance labels should be far from each other. Mathematically, this objective is achieved by defining a loss function:

$$L_{ins} = L_{intra} + L_{inter} + L_{reg}, \tag{2}$$

which is composed of the intra-point loss L_{intra}, inter-point loss L_{inter}, and regularization loss L_{reg}.

The intra-point loss L_{intra} is defined by

$$L_{intra} = \frac{1}{M}\sum_{m=1}^{M}\frac{1}{N_m}\sum_{i=1}^{N_m}[\|\mu_m - C_i\| - \delta_v]_+^2, \tag{3}$$

where M represents the number of instances, and N_m is the number of points belonging to instance m. The intra-point loss encourages the embedding points C_i of the same instance to be close to the embedding instance center μ_m. The margin δ_v softly constrains the embedding point to be within a certain range δ_v from the center C_i. The notation $[n]_+$ is a function that computes $max(0, n)$.

The inter-point loss L_{inter} aims to push the centers of two different instances, μ_m^A and μ_m^B, away as far as possible. Similarly, the margin term of inter-point loss forces the different instance centers to be away at most $2\delta_d$,

$$L_{inter} = \frac{2}{M(M-1)}\sum_{A=1}^{M}\sum_{B=1,B\neq A}^{M}[2\delta_d - \|\mu_m^A - \mu_m^B\|]_+^2. \tag{4}$$

The regularization loss L_{reg} encourages more compact and centralized centers. Through minimizing the norm of centers, L_{reg} ensures that all centers are close to the origin of the feature space.

$$L_{reg} = \frac{1}{M}\sum_{m=1}^{M}\|\mu_m\| \tag{5}$$

D. Loss function

The total loss L_{total} in our model is the weighted sum of the upsampling loss and the auxiliary instance embedding loss:

$$L_{total} = \alpha L_{up} + \beta L_{ins}. \tag{6}$$

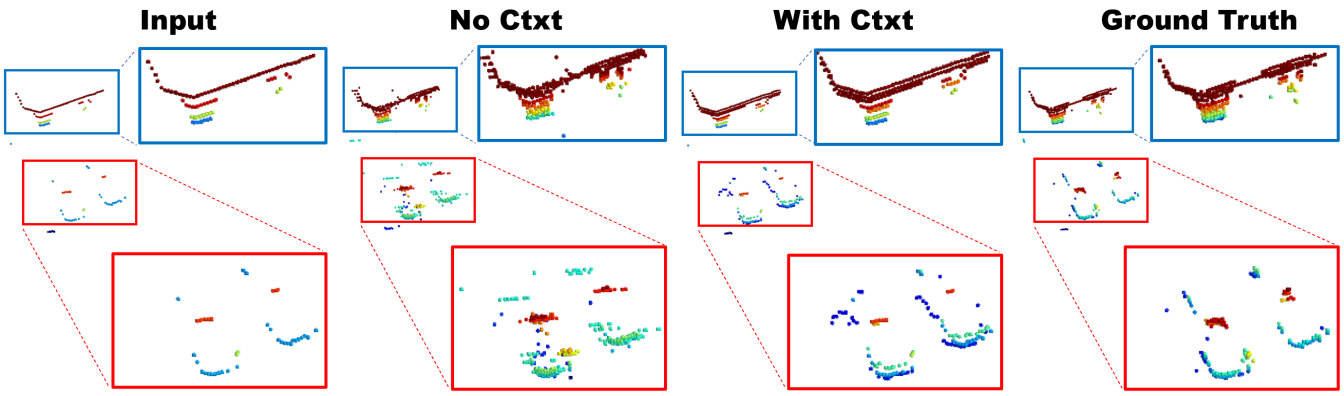

Fig. 3: An example of a region distant from the LiDAR center reveals incomplete and sparse structures for both buildings and cars. No Ctxt and with Ctxt represent the model with and without context attention module, respectively. With the context attention module, irrelevant points are ignored, resulting in a more accurate shape that better preserves contours and is more similar to the Ground Truth.

In our experiments, we set α to 100 and β to 1 empirically. Chamfer distance is adopted as the upsampling loss. It measures the differences between the upsampled point cloud P and ground truth Q:

$$L_{up} = \frac{1}{|P|}\sum_{p\in P}\min_{q\in Q}\|p-q\|_2^2 + \frac{1}{|Q|}\sum_{q\in Q}\min_{p\in P}\|p-q\|_2^2, \quad (7)$$

where p and q denote the 3D position of a point in P and Q, respectively.

IV. Experimental Results

A. Dataset preparation

Our experiment is conducted on the SemanticKITTI [1]. We utilize annotations from the panoptic segmentation task to train our instance embedding auxiliary task. Specifically, among the total 19 classes, 8 classes are the foreground objects, termed "thing classes", while the other 11 classes are background elements, termed "stuff classes", in the SemanticKITTI definition. Points belonging to "thing classes" are assigned instance IDs to form distinct instance groups, while "stuff classes" only have semantic labels without instance IDs. In our instance embedding training setting, we treat all points within the same "stuff class" as the same instance group, which helps the network to better distinguish between foreground objects and background elements.

The training dataset is constructed using pairs of 32-beam and 64-beam LiDAR scenes. We followed the downsampling method mentioned in [27] to generate the 32-beam scenes. Sequence 00 is used as the training sequence. To encourage our model INS-ATT to infer the instance relationships, we first find the points with the same instance ID and determine the patch center, which is set as the center of each instance. Each patch may contain points with multiple instance IDs, consisting of the main instance (the ID for the patch center) and sub-instances (other ID points). As a result, a single instance ID can appear repeatedly in several patches, but among all these patches, it will be the main instance for only one of them. Then, all points within 5 meters of the instance center are retained as the patches. Repeating the same steps, we obtain patches in the 64-beam dataset and form 73,000 pairs of 32-beam and 64-beam patches for training.

Please note that instance embedding is only used during the training process, and is removed during testing. This means that we do not require labels during the testing phase. The Farthest Point Sampling algorithm [24] is used to obtain the patch centers. After all the patches are upsampled by the model, the patches are first combined together and a certain number of points are sampled using the Farthest Point Sampling. In the following evaluation experiments, the upsampling ratio is set to be 2x.

B. Evaluation metric

We followed [17], [18] to use Voxel Intersection over Union (Voxel IoU), precision, recall and F1-score to evaluate the 3D upsampling quality in LiDAR scene. Voxel IoU is defined as:

$$IoU = \frac{|TP|}{|TP|+|FP|+|FN|} \quad (8)$$

where True Positive (TP) indicates the simultaneous presence of voxels in both the upsampled and ground truth scenes; False Positive (FP) denotes the presence of voxels in the upsampled scene but not in the ground truth scene, while False Negative (FN) represents the opposite scenario of FP. All the metrics are evaluated with 0.1m grid size.

C. Results on entire LiDAR scenes

We compared the proposed method with NPT [4], PU-GCN [5], and MPU [6]. The settings of NPT are described in Section III-A. The experiment was conducted on sequences 01 to 10 of the SemanticKITTI dataset. For each sequence, 100 frames were randomly selected as the target frames for the models to infer the upsampling scene. All the comparison methods were retrained on our selected LiDAR pair dataset from SemanticKITTI. Table I lists the results of each method. Compared to the other three methods, our method INS-ATT achieved the highest Voxel IoU, surpassing the others by approximately 4% to 10%. Additionally, our method achieved the best precision of 0.538 while maintaining a close recall value of 0.661 to the best recall 0.669. It indicates that our methods effectively reduces artifacts in the scene. Figure 4 shows a visual comparison of the upsampling results. It's evident that our upsampling results (INS-ATT) exhibit clear

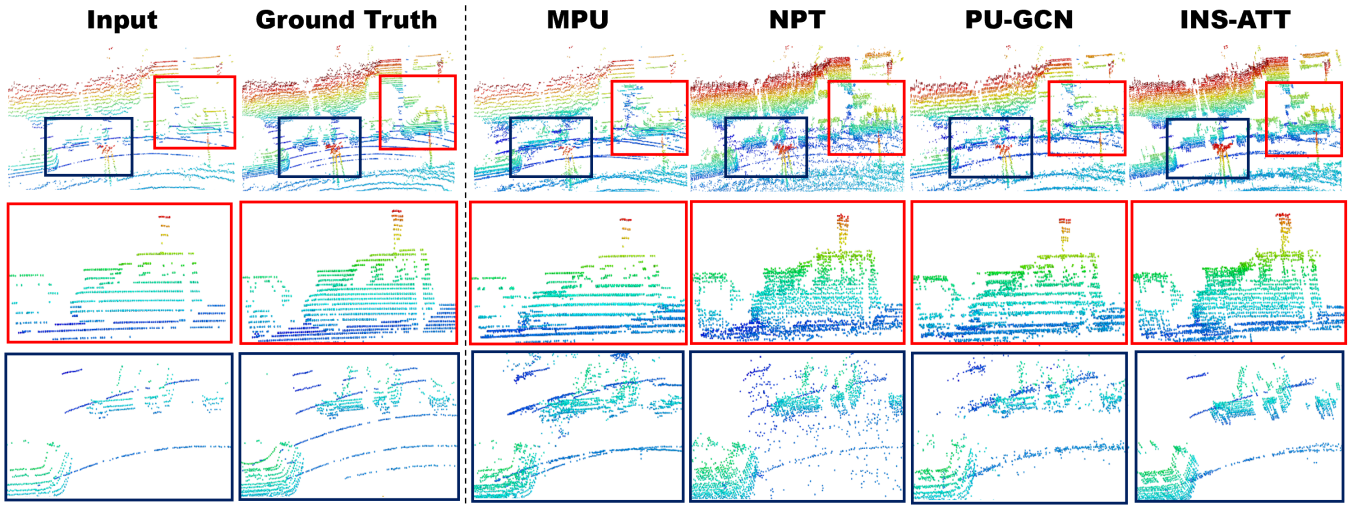

Fig. 4: The results of LiDAR upsampling via different models. The highlighted regions inside the boxes are enlarged and shown in the 2nd and 3rd rows. Our method, INS-ATT, reduces the artifacts in the upsampling result and presents clear boundary of the instances.

boundaries, with most upsampling points generated on the instance surface.

	IoU	Precision	Recall	F1-score
Whole scene				
INS-ATT	**0.421**	**0.538**	0.661	**0.592**
[4] NPT	0.329	0.393	**0.669**	0.494
[5] PU-GCN	0.389	0.511	0.618	0.559
[6] MPU	0.353	0.503	0.543	0.521

TABLE I: Whole-scene results on SemanticKITTI.

D. Results on object instances

By incorporating an instance embedding auxiliary branch, we anticipate improved performance across various instance classes. Enhancing the density of each instance also contributes to more effective 3D detection, making the quality of instance upsampling a critical factor. In this section, we further evaluate the network's ability to reconstruct fine details by assessing the Voxel IoU of sub-scenes surrounding the instances.

Car, person, and cyclist are the three primary instance classes in the SemanticKITTI dataset, and they are crucial objects for detection in self-driving cars. They were chosen as the target instance classes for evaluation. To capture the area surrounding each instance, we first identify the point set for the instance class and define a cuboid as its boundary. The cuboid's size is determined by the maximum and minimum coordinate values of the point set. Additionally, to consider the artifacts surrounding each instance, points outside the cuboid but within a 0.3m range are also collected. Table II presents the Voxel IoU results for car, person, and cyclist, respectively. Our method achieves the best Voxel IoU, recall and F1-score across all three instance classes. Compared to the second-best method, our proposed method on the Voxel IoU shows a significant improvement of about 8% on the car class, approximately 5% on the person class, and around 6% on the cyclist class.

E. Ablation study

We validate the effectiveness of the context attention module and the instance embedding auxiliary branch. The

	IoU	Precision	Recall	F1-score
Car				
INS-ATT	**0.373**	**0.554**	**0.579**	**0.530**
[4] NPT	0.231	0.277	0.575	0.361
[5] PU-GCN	0.287	0.386	0.510	0.427
[6] MPU	0.292	0.441	0.492	0.431
Person				
INS-ATT	**0.405**	0.571	**0.620**	**0.563**
[4] NPT	0.270	0.321	0.609	0.408
[5] PU-GCN	0.318	0.421	0.540	0.459
[6] MPU	0.358	**0.599**	0.495	0.505
Cyclist				
INS-ATT	**0.393**	**0.557**	**0.610**	**0.553**
[4] NPT	0.287	0.345	0.606	0.432
[5] PU-GCN	0.335	0.443	0.534	0.479
[6] MPU	0.303	0.435	0.508	0.450

TABLE II: Object instance results on SemanticKITTI: Car, Person, and Cyclist.

baseline is NPT, as mentioned in Section III-A. "+Ctxt" and "+Ins" denote the baseline with the addition of context module and instance embedding auxiliary branch, respectively.

Table III shows the ablation results of the whole scene. Adding the context module boosts the whole scene Voxel IoU by about 5%. Similarly, the additional instance embedding auxiliary task improves the Voxel IoU by 6%. Both components contribute to the improvement in the upsampling result. This indicates that context refinement on the local feature and being aware of the instance relationships benefit the upsampling quality.

We further evaluate the upsampling quality around the major instance class in Table IV. Compared to "+Ctxt", "+Ins" achieves significantly higher improvements while maintaining a close recall across all three classes. This improvement demonstrates that instance-aware learning provides greater benefits when focusing on detailed construction in close proximity to instances within the scene. The slight decrease in recall when both modules are used can be attributed to the model becoming more conservative in generating points between objects. That is, incorporating both contextual and instance information encourages the model to reduce point generation in ambiguous regions between objects, potentially

leading to fewer points in these areas. This trade-off explains why our method produces fewer artifacts compared to other 3D upsampling methods on point clouds, as it could prioritize accuracy over completeness in challenging regions.

	IoU	Precision	Recall	F1-score
Whole scene				
[4] Baseline	0.329	0.393	0.669	0.494
+ Ctxt	0.372	0.446	0.690	0.541
+ Ins	0.388	0.465	**0.702**	0.559
+ Ctxt + Ins	**0.421**	**0.538**	0.661	**0.592**

TABLE III: Ablation study results of whole scenes on SemanticKITTI.

	IoU	Precision	Recall	F1-score
Car				
[4] Baseline	0.231	0.277	0.575	0.361
+ Ctxt	0.274	0.334	0.608	0.418
+ Ins	0.358	0.497	**0.609**	0.511
+ Ctxt + Ins	**0.373**	**0.554**	0.579	**0.530**
Person				
[4] Baseline	0.270	0.321	0.609	0.408
+ Ctxt	0.306	0.360	**0.660**	0.454
+ Ins	0.367	0.472	0.652	0.521
+ Ctxt + Ins	**0.405**	**0.571**	0.620	**0.563**
Cyclist				
[4] Baseline	0.287	0.345	0.606	0.432
+ Ctxt	0.326	0.387	**0.667**	0.481
+ Ins	0.383	0.471	0.656	0.534
+ Ctxt + Ins	**0.393**	**0.557**	0.610	**0.553**

TABLE IV: Ablation study results of Car, Person and Cyclist classes on SemanticKITTI.

V. Conclusion

In this paper, we introduce INS-ATT, an instance-aware LiDAR scene upsampling network. We incorporate an instance embedding auxiliary task, along with the main upsampling task, during training. This auxiliary task is crucial for capturing instance-level information in sparse LiDAR scenes, enhancing the network's ability to reduce artifacts near instances by leveraging instance priors encoded in the shared features. Additionally, we introduce a context attention module into the feature extraction phase. This module enriches the representation by combining local and contextual information, refining the attention mechanism to enhance shape accuracy in sparser areas and minimize the influence of irrelevant points. Experimental results substantiate the effectiveness of our method in generating fewer artifacts in real LiDAR scenes. In the future, we aim to explore replacing instance priors with clustering or contrastive learning to evaluate their impact on model performance while reducing the need for instance annotations. Further exploration may also investigate the potential for applying instance loss to enhance the effectiveness of other methods.

References

[1] J. Behley, M. Garbade, A. Milioto, J. Quenzel, S. Behnke, C. Stachniss, and J. Gall, "SemanticKITTI: A dataset for semantic scene understanding of LiDAR sequences," in *ICCV*, October 2019.

[2] A. X. Chang, T. A. Funkhouser, L. J. Guibas, P. Hanrahan, Q. Huang, Z. Li, S. Savarese, M. Savva, S. Song, H. Su, J. Xiao, L. Yi, and F. Yu, "ShapeNet: An information-rich 3D model repository," *CoRR*, vol. abs/1512.03012, 2015.

[3] Z. Wu, S. Song, A. Khosla, F. Yu, L. Zhang, X. Tang, and J. Xiao, "3D ShapeNets: A deep representation for volumetric shapes," in *CVPR*, June 2015.

[4] W. Feng, J. Li, H. Cai, X. Luo, and J. Zhang, "Neural Points: Point cloud representation with neural fields for arbitrary upsampling," in *CVPR*, June 2022, pp. 18 633–18 642.

[5] G. Qian, A. Abualshour, G. Li, A. Thabet, and B. Ghanem, "PU-GCN: Point cloud upsampling using graph convolutional networks," in *CVPR*, June 2021, pp. 11 683–11 692.

[6] W. Yifan, S. Wu, H. Huang, D. Cohen-Or, and O. Sorkine-Hornung, "Patch-based progressive 3D point set upsampling," in *CVPR*, June 2019.

[7] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "PointNet: Deep learning on point sets for 3D classification and segmentation," in *CVPR*, July 2017.

[8] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, "PointNet++: deep hierarchical feature learning on point sets in a metric space," in *NeurIPS*, 2017, p. 5105–5114.

[9] A. V. Phan, M. L. Nguyen, Y. L. H. Nguyen, and L. T. Bui, "DGCNN: A convolutional neural network over large-scale labeled graphs," *Neural Networks*, vol. 108, pp. 533–543, 2018.

[10] D. Xu, A. Vedaldi, and J. F. Henriques, "Moving SLAM: Fully unsupervised deep learning in non-rigid scenes," in *IROS*, 2021, pp. 4611–4617.

[11] S. Liu, A. Davison, and E. Johns, "Self-supervised generalisation with meta auxiliary learning," in *NeurIPS*, 2019.

[12] A. Navon, I. Achituve, H. Maron, G. Chechik, and E. Fetaya, "Auxiliary learning by implicit differentiation," in *ICLR*, 2021.

[13] L. Yu, X. Li, C.-W. Fu, D. Cohen-Or, and P.-A. Heng, "PU-Net: Point cloud upsampling network," in *CVPR*, June 2018.

[14] R. Li, X. Li, C.-W. Fu, D. Cohen-Or, and P.-A. Heng, "PU-GAN: A point cloud upsampling adversarial network," in *ICCV*, 2019.

[15] T. Shan, J. Wang, F. Chen, P. Szenher, and B. Englot, "Simulation-based lidar super-resolution for ground vehicles," *Robotics and Autonomous Systems*, vol. 134, p. 103647, 2020.

[16] Y. Jung, S.-W. Seo, and S.-W. Kim, "Fast point clouds upsampling with uncertainty quantification for autonomous vehicles," in *ICRA*, 2022, pp. 7776–7782.

[17] Y. Kwon, M. Sung, and S. Yoon, "Implicit LiDAR network: LiDAR super-resolution via interpolation weight prediction," in *ICRA*, 2022, pp. 8424–8430.

[18] M. Park, H. Son, and E. Kim, "Implicit point function for LiDAR super-resolution in autonomous driving," *IEEE Robotics and Automation Letters*, vol. 8, no. 11, pp. 7003–7009, 2023.

[19] T.-Y. Chen, C.-C. Hsiao, and C.-C. Huang, "Density-imbalance-eased LiDAR point cloud upsampling via feature consistency learning," *IEEE Transactions on Intelligent Vehicles*, vol. 8, no. 4, pp. 2875–2887, 2023.

[20] A. Savkin, Y. Wang, S. Wirkert, N. Navab, and F. Tombari, "Lidar upsampling with sliced wasserstein distance," *IEEE Robotics and Automation Letters*, vol. 8, no. 1, pp. 392–399, 2023.

[21] S. Kolouri, P. E. Pope, C. E. Martin, and G. K. Rohde, "Sliced wasserstein auto-encoders," in *ICLR*, 2019.

[22] J. Wu, Z. Huang, D. Acharya, W. Li, J. Thoma, D. P. Paudel, and L. V. Gool, "Sliced wasserstein generative models," in *CVPR*, June 2019.

[23] K. Nguyen, N. Ho, T. Pham, and H. Bui, "Distributional sliced-wasserstein and applications to generative modeling," in *ICLR*, 2021.

[24] Y. Eldar, M. Lindenbaum, M. Porat, and Y. Zeevi, "The farthest point strategy for progressive image sampling," *IEEE Transactions on Image Processing*, vol. 6, no. 9, pp. 1305–1315, 1997.

[25] X. Pan, Z. Xia, S. Song, L. E. Li, and G. Huang, "3D object detection with Pointformer," in *CVPR*, June 2021, pp. 7463–7472.

[26] B. D. Brabandere, D. Neven, and L. V. Gool, "Semantic instance segmentation with a discriminative loss function," 2017.

[27] D. Zermas, I. Izzat, and N. Papanikolopoulos, "Fast segmentation of 3D point clouds: A paradigm on lidar data for autonomous vehicle applications," in *ICRA*, 2017.