IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2024, Yokohama, Japan. Cite as RA-L paper.

Dynamic Object Classification of Low-resolution Point Clouds: An LSTM-based Ensemble Learning Approach

Shaoming Zhang[1], *Member, IEEE*, Tangjun Yao[1], Jianmei Wang[1]*, Tiantian Feng[1], and Zhong Wang[2]

***Abstract*—In unmanned vehicle perception, dynamic object classification is applied to classify objects accurately and timely, providing decision-making for obstacle avoidance and planning. Low-resolution LiDAR is one of the most important sensors for this task. Unfortunately, the existing approaches perform unsatisfactorily due to the huge domain gap between low-resolution and high-resolution point cloud classification. Some schemes try to reduce the gap by fusing multi-scan information through SLAM or completing single-scan point clouds. However, these methods rely on high positioning accuracy or the wholeness of object data. To this end, differently, we propose a dynamic object classification method of low-resolution data from the perspective of time-series fusion. By modeling time series of sparse data, we indicate change rules of separate classification models for object representation. Subsequently, based on ensemble learning, our method performs feature-level fusion on multiple networks to exploit their different expression capabilities. Finally, we utilize long short-term memory to gradually classify dynamic objects. Besides, we also propose a dataset of the low-resolution point clouds and manually annotate the ground truth, which contains abundant samples of cars, pedestrians, and motorcycles. Through testing actual low-resolution data, the accuracy of our method is verified to improve a lot than the state-of-the-art approaches.**

***Index Terms*—Recognition, Computer Vision for Automation, Low-resolution point clouds, ensemble learning, long short-term memory.**

I. Introduction

LOW-SPEED unmanned vehicles show great potential of application in industrial assembly, logistics delivery, cleaning sanitation, etc. In relevant applications, thanks to its advantages of accurate distance measurement, strong anti-interference, long working time and high cost performance, the low-resolution 16-beam LiDAR is widely used for dynamic object perception. As the main research module of dynamic object perception, point cloud classification aims to accurately classify obstacles around unmanned vehicles in real time and warn of possible dangers. The existing methods of classification consist of the traditional ones and the deep learning-based ones. The former approaches depend on handcraft features, for example, the geometry-based, the statistic-based and the motion-based [2], [3], [4]. Such approaches are efficient and explainable, while their features are limited in certain scenes because of relatively low representational ability. The latter ones can be categorised into three types, graph-based [15], [16], convolution-based [7], [10]–[12], and point-based [19], [20]. They show pleasing classification results on high-resolution point cloud datasets (such as ModelNet [5], KITTI [6], etc.). However, when such methods are transferred to the classification of 16-beam low-resolution point cloud, even after retraining or fine-tuning, it is still difficult to obtain satisfactory results. The underlying reason is that there is actually a huge domain gap in object classification between low-resolution and high-resolution point clouds. As shown in Fig. 1, the scanned patterns of objects (car, pedestrian) in a low-resolution LiDAR and a high-resolution LiDAR are significantly different. Therefore, how to achieve accurate object classification of low-resolution point clouds is a considerably challenging problem to be solved.

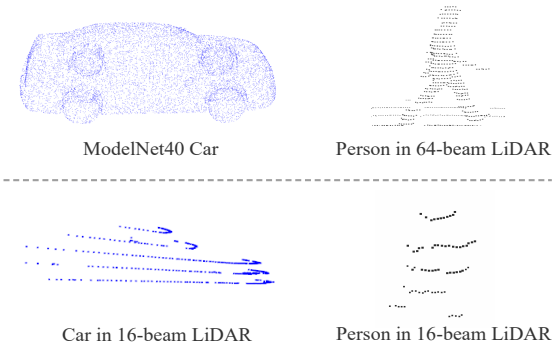

Fig. 1. Showcasings of dynamic objects data from different datasets. *Top:* Point clouds of a car from ModelNet [5] and a pedestrian from KITTI [6] which uses Velodyne HDL-64E. *Bottom:* Data obtained by RS-LiDAR-16. Sparsity of point clouds makes it challenging for dynamic objects classification of low-resolution LiDAR data.

To tackle this problem, a natural idea is to generate dense point clouds from their corresponding sparse ones. To date, there are two main types of schemes to fulfill this purpose. Some works utilize real-time pose information of the carriers given by Simultaneous Localization and Mapping (SLAM) and combine the previous point clouds for data fusion. In such a way, although the object point clouds become more detailed, when the poses given by SLAM have small errors, it can easily lead to the matching failure between dynamic objects and cause subsequent classification errors. Another type of

Manuscript received: June, 5, 2023; Revised August, 7, 2023; Accepted October, 5, 2023.

This paper was recommended for publication by Editor V. Markus upon evaluation of the Associate Editor and Reviewers' comments. This work was supported by the National Key RD Program of China (2021YFB2501103).

[1]Shaoming Zhang, Tangjun Yao, Jianmei Wang, and Tiantian Feng are with the College of Surveying and Geo-Informatics, Tongji University, Shanghai, 200092, China (e-mail: zhangshaoming@tongji.edu.cn; 2111331@tongji.edu.cn; jianmeiw@tongji.edu.cn; fengtiantian@tongji.edu.cn).

[2]Zhong Wang is with the College of Software Engineering, Tongji University, Shanghai 201804, China (e-mail: 2010194@tongji.edu.cn).

Digital Object Identifier (DOI): see top of this page.

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2024, Yokohama, Japan. Cite as RA-L paper.

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2024, Yokohama, Japan. Cite as RA-L paper.

methods try to interpolate points into the single low-resolution scan [23], [24]. Although they can complete a certain local point set of 16-beam LiDAR data and get a relatively detailed object profile, the correctness of the interpolated point clouds remains to be verified. Therefore, robustness and accuracy of the present schemes remain to be improved in object classification of low-resolution point clouds.

Different from the aforementioned idea of directly completing sparse point clouds, we consider narrowing the domain gap between the object classification of low-resolution and high-resolution point clouds from the perspective of time-series fusion. On the basis of that, we propose a low-resolution point cloud classification method based on Long Short-Term Memory (LSTM) and ensemble learning, termed **PointLE**. Specifically, our motivation is that although the existing neural networks are not satisfactory for single scan dynamic object classification, it is able to classify objects according to the change of their high-dimensional features in a certain period of time. Also, different networks encode separate high-dimensional features of dynamic objects. Therefore, PointLE adopts the time-series fusion method similar to the SLAM-based ones to encode multi-scan information, but does not rely on accurate poses to avoid point cloud registration error. Instead, it exploits the idea of ensemble learning to output the high-dimensional feature variations of dynamic objects in a time sequence from different networks. Afterwards, we fuse point clouds at the feature level to form deep information of objects in the time series. Subsequently, we resort to LSTM to decode this time series data and gradual classification is carried out according to the change of high-dimensional features of dynamic objects. Moreover, in the actual classification, due to the similar point cloud distribution of pedestrians and motorcyclists in the distance, it is hard to distinguish them in this case. Considering that there is a certain gap between the moving speeds of these two types of objects, we also take the change of distance between the LiDAR and the dynamic objects in the time series as one of the input of the LSTM. Through the above designs, PointLE effectively integrates the time-series information of point clouds and reduces the field difference between low-resolution and high-resolution point cloud classification. In this way, the robustness and accuracy of point cloud classification are improved in actual scenes. Since there is no open source dataset of 16-beam LiDAR for low-resolution point cloud classification, we propose a new point cloud dataset collected in Tongji University which includes abundant annotated cars, pedestrians, and motorcycles.

To summarize, our contributions are twofold:

1) We propose PointLE, an object time-series fusion method based on ensemble learning, which fully makes use of different representation capabilities of various models for sparse point clouds to encode data at the feature level. Subsequently, we fulfill gradual object classification through LSTM, reducing the field difference between low-resolution and high-resolution point cloud object classification.
2) We provide a dataset of dynamic objects collected by a low-speed unmanned vehicle equipped with 16-beam LiDARs, which is manually annotated and double checked. Our PointLE is fully evaluated on this dataset. The results show that PointLE is superior to the existing classification methods in the field of low-resolution point cloud object classification, achieving an overall accuracy of 98.73%. To make our results fully reproducible, our dataset[1] and the relevant codes[1] are open-sourced online.

The remainder of this paper is organized as follows. Section II introduces related works. Section III makes an overview of the pipeline of PointLE and introduces the methodology in detail. Section IV presents the experiments of the public and collected datasets. Finally, Section V concludes the paper.

II. Related Work

A. Point Cloud Classification Methods

In recent years, researchers have proposed a variety of point cloud classification methods, which can be divided into traditional methods and deep learning methods. The traditional ones include the template matching-based approaches and the handcraft feature-based approaches. In [1], point clouds are projected into two-dimensional grids. Afterwards, objects in the scene are found according to the established template library. Considering the diversity of dynamic objects and the existence of occlusion in the actual scene, the establishment of template library is actually difficult and the matching is impractical. Differently, the handcraft feature-based approach [3] aims to extract the geometric, statistical and motional features of the target and utilizes machine learning discriminator for classification. Despite these features are explicable and computationally efficient, their representation remains to be improved when faced with complex classification tasks.

In view of this reason, the current research resorts to deep learning networks to obtain more informative features. To date, deep learning classification methods are mainly divided into three categories: the graph-based, the CNN-based and the point-based. In order to benefit from the mature field of image processing, the graph-based approach MVCNN [15] transforms 3D point clouds into a set of multiple views, utilizing the view-pooling operational classification. In a different technology roadmap, DGCNN [16] is based on the feature space. Its graph structure formed in each layer constantly changes to better capture the local relations of the feature dimensions. As for the disadvantage, the deep features within the neighbourhood share high similarities so that edges of this structure are ineffective. Instead of processing images with multiple views, the CNN-based approach depends on the outstanding ability of CNN to learn geometric relationships between voxels. Specifically, VoxNet [7] represents the point cloud structure via occupation grids, followed by the convolution neural network for classification. In such way, the operation of directly voxerizing disordered point clouds leads to loss of neighborhood relations between point clouds, undermining the final classification results. Different from the direct way of using 3D occupation grids, 3DmFV [12] describes points by deviations from the Gaussian mixture model, which maintains the continuity of point clouds. The point-based approaches directly process points, which means

[1]https://github.com/ytj-dasd/PointLE

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2024, Yokohama, Japan. Cite as RA-L paper.

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2024, Yokohama, Japan. Cite as RA-L paper.

that they keep more details than those which divide point clouds into voxels. For example, PointNet [19] fulfills classification by means of the global feature extracted from the raw point clouds. Considering PointNet [19] ignoring local neighborhood information of point clouds, PointNet++ [20] extracts local features of the target at first. To carry out that, PointNet++ [20] requires multiple grouping and sampling, thus the computational efficiency needs to be improved. The above-mentioned point cloud networks aim at detailed datasets such as ModelNet [5] and 64-beam LiDAR, demonstrating accurate classification. But when such networks are applied to the classification of low-resolution point clouds, the results are far below expectation.

B. Domain Adaptation in Point Cloud

In order to narrow the domain gap between low-resolution and high-resolution data, the explicit and implicit methods have emerged recently. The former ones aim at completing the low-resolution data, which are classified into the multiple scan-based ones and the single scan-based ones. According to the poses given by SLAM, Zhang *et al.* [2] use ICP to combine data from multiple scans, obtaining dynamic objects with more detailed point clouds. However, when the poses have centimeter-level deviation, the stability of the ICP method drops sharply, making the fusion effect unsatisfactory. To generate dense point clouds from a single scan, the second type of the methods [24] realizes the structure in a coarse-to-fine way. On this basis, [23] proposes layer-by-layer completion by multi-resolution pyramid combined with generative adversarial network (GAN), aiming to smoothly generating point clouds. However, this method is mostly applied to the case where only a part of the target containing dense point cloud is blocked. Due to the sparse data of the low-resolution LiDAR, the object profile obtained by 16-beam LiDAR scan is highly insufficient, thus the ideal completion cannot be obtained.

The implicit methods adapt the model pre-trained from the source domain to the target domain. In order to avoid the large burden of 3D annotation during the adaptation, ST3D [38] adopts self-training to achieve 3D unsupervised domain adaptation (UDA) by enhancing pseudo labels. Despite these efforts, there is still an obvious gap between the UDA and fully-supervised approaches. To deal with it, the semi-supervised methods attempt to select a small portion of labeled samples for training [37], [39]. For example, Bi3D [39] utilizes the idea of active learning to select samples from two aspects. Specifically, it extracts samples similar to the targets in the source domain and conducts a diversity-based sampling strategy in the target domain. In such way, improvements have been made in adapting to LiDAR with the same or higher beams. However, the accuracy of cross-domain adaptation from 64-beam to 32-beam is still relatively unsatisfactory. Considering this issue, SSDA3D [36] learns domain-invariant knowledge and better generalizes it into the target domain through a two-stage approach. In fact, the difficulty of adapting from ModelNet40 to the 16-beam domain is greater than what SSDA3D [36] has done. Compared with the above methods, PointLE utilizes high-resolution domain features at a deeper level by combining ensemble learning and temporal fusion, obtaining the best accuracy in low-resolution data.

C. Sequential Data Processing Methods

To fuse information of dynamic objects, we regard point clouds in multiple scans as sequential data for gradual classification. By exploiting the sequential information, several methods have been proposed to address the multiple-scan 3D object perception in recent years. PointRNN [31] utilizes RNNs to predict scene flow on multi-scan point clouds. By means of the neighbour search based on point coordinates, PointRNN [31] aggregates the previous state and current input, aiming to predict the future trajectories of points. Instead of making considerable efforts into finding nearest points, Huang *et al.* [32] adopt 3D sparse convolution to extract features in a scan. Subsequently, these features are fed into LSTM [25] together with the hidden and memory information for multi-scan fusion. By continuously updating long-term information, LSTM [25] is able to constantly improve the confidence of the object category based on the sequential input. Another work [33] divides sequential information into short-term and long-term data in order to boost the 3D object detector. The short-term one is used to model the grid-level features of fused scans through iterations. The long-term one focuses on detecting small objects and aligning dynamic objects which are challenging during the process. In our work, we also resort to LSTM [25] due to its capability of gradual classification. Differently, we take the high-dimensional features of objects as the criteria for LSTM [25], avoiding aligning tasks.

III. Methodology

As shown in Fig. 2, our PointLE consists of three parts. The first part removes ground point cloud attached to the dynamic objects by the plane segmentation algorithm. The second part adopts the idea of ensemble learning to utilize three deep learning classification networks: PointNet++ [20], 3DmFV [12] and DGCNN [16], and form feature vectors. The third part combines these features with previous moments to form sequential data. Meanwhile, the input of PointLE considers the object distance from the LiDAR, which is fed into LSTM with the feature vectors to achieve gradual classification.

A. Ground Segmentation

As shown in Fig. 3, during the movement of object, data obtained by LiDAR not only includes the object point clouds, but also contains some ground points. Due to the sparsity of object data, the attached ground points can easily interfere with the classification, making it more challenging. In our approach, ground point clouds at the bottom of dynamic objects are removed by establishing the ground model in the scene. The main process of ground extraction is divided into three parts. Firstly, point cloud is transformed from the original cartesian coordinate representation to representation of the azimuth angle, vertical angle and polar distance. Afterwards, we resort to Line-Fit [27] for each sector region to obtain the initial ground skeleton points. Finally, assuming that the ground point elevation in the sector region satisfies the gaussian distribution, the above ground skeleton points are used as training data to

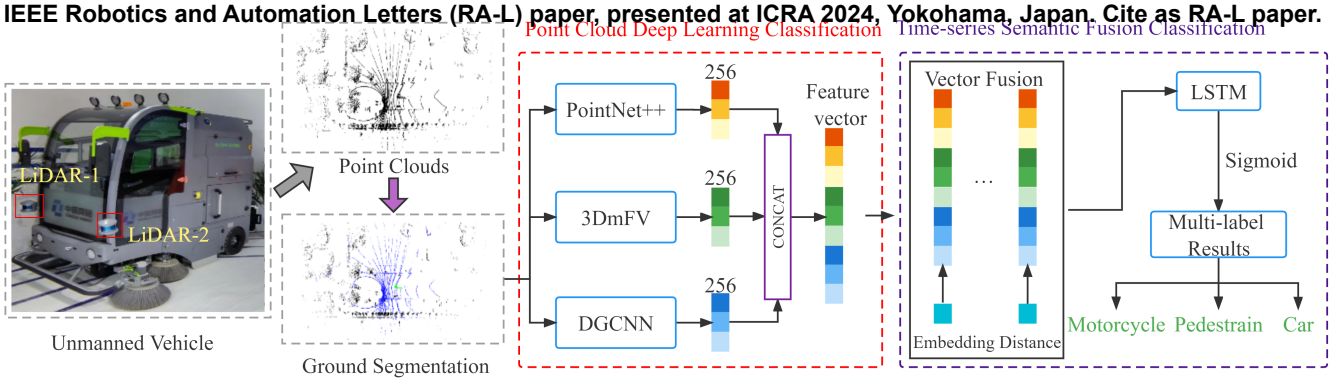

Fig. 2. The architecture of our proposed PointLE. Two 16-beam LiDARs are installed on the left and right side of an unmanned vehicle to collect point clouds. The input ground points are removed through the ground segmentation part. Next, the 256-dimensional features are extracted from deep learning networks to form the feature vector. By connecting embedded distances with this feature vector, the last part uses LSTM to fulfill time-series semantic fusion classification.

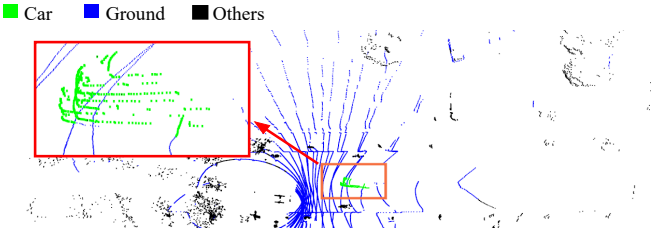

Fig. 3. The result of ground extraction. The car is labeled in green, the ground points in blue, and the others in black.

build the ground Gaussian process model. This method is able to represent complex ground situations better than only using Line-Fitting. As an example, Fig. 3 illustrates the experimental results after the ground extraction.

B. Ensemble Learning

Although existing point cloud classification networks cannot provide a satisfactory result for sparse point clouds, we believe that the extracted features of networks are still helpful for dynamic object classification. We examine the abilities of different network models to identify various types of objects. Table I indicates classes that have a high frequency of appearance in classification of dynamic objects, from which it can be seen that these networks concern different point cloud features. For example, PointNet++ [20] focuses its feature extraction on regular geometric classes, such as curtains, tents, and cones, while PointConv [11] outputs semantic labels that are mostly complex and irregular categories, such as guitars, airplanes, and lamps. In terms of pedestrian classification, 3DmFV [12] maintains the continuous properties of the point clouds. For this reason, such network captures local details of pedestrians relatively well. Moreover, RSCNN [10] and PointConv [11] is able to learn the spatial topology of pedestrian point clouds for classification. As for motorcycle, PointNet++ [20] and DGCNN [16] still show classification abilities although the motorcycle data in the dynamic scene are extremely sparse.

Therefore, we exploit the idea of ensemble learning to integrate multiple networks at the feature level, fusing the representation abilities of different models. Specifically, we exploit 256-dimensional features of the fully connected layer in each backbone. After the feature extraction, we adopt the feature-level fusion to integrate the object information. The specific method is to form a feature vector which combines the

TABLE I

SEVERAL CLASSES WITH HIGH FREQUENCY IN THE DYNAMIC OBJECT CLASSIFICATION. THE RESULTS ARE OBTAINED BY UTILIZING THE BASELINES TO PREDICT OUR 16-BEAM DATA. THEY ARE SORTED BY FREQUENCY IN DESCENDING ORDER. CORRECT RESULT IN BOLD.

	PointNet++	3DmFV	DGCNN	RSCNN	PointASNL	PointConv
Car	curtain	**car**	**car**	guitar	guitar	guitar
	tent	tent	plant	lamp	**car**	**car**
	bowl	lamp	bowl	radio	airplane	radio
	keyboard	piano	bed	**car**	motorcycle	lamp
	cone	pedestrian	guitar	tent	radio	airplane
Pedestrian	plant	**pedestrian**	plant	lamp	plant	**pedestrian**
	cone	plant	car	vase	bookshelf	plant
	keyboard	lamp	motorcycle	plant	**pedestrian**	lamp
	curtain	stairs	bookshelf	guitar	flowerpot	curtain
	flowerpot			flowerpot	lamp	
Motorcycle	plant	pedestrian	plant	plant	plant	plant
	flowerpot	plant	car	pedestrian	pedestrian	pedestrian
	cone	stairs	**motorcycle**	flowerpot	stairs	stairs
	keyboard	car		lamp	car	flowerpot
	motorcycle	lamp				

768-dimensional feature. By comparing with another strategy of representing object features, the high-dimensional feature-based one is verified to be the best in the supplementary[1].

C. Time-series Fusion

After the feature concatenation, we conduct a time-series fusion to integrate the object information in 1.5 seconds. The specific method is to form a sequence vector which contains all of the 768-dimensional features obtained in 1.5 seconds.

As shown in Fig. 4, when the distance between the motorcyclist and the LiDAR is far away, point clouds of the motorcycle are much less than that of the pedestrian. Consequently, pedestrians and motorcyclists in the distance cannot be classified. Considering that the distance between dynamic objects and the LiDAR is an important factor, we try to incorporate it into time-series fusion network through two methods. The purpose of the former method is to find the boundary of the distance between the LiDAR and objects. When the distance exceeds the threshold, we only divide target into pedestrian and car. Since we focus on the low-speed unmanned driving scene, robot has sufficient decision time. When the distance is less than the threshold, the dynamic object is still finely divided into car, pedestrian and motorcycle. The latter idea is to regard the distance as the input of the LSTM along with 768-dimensional vector extracted from three backbones.

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2024, Yokohama, Japan. Cite as RA-L paper.

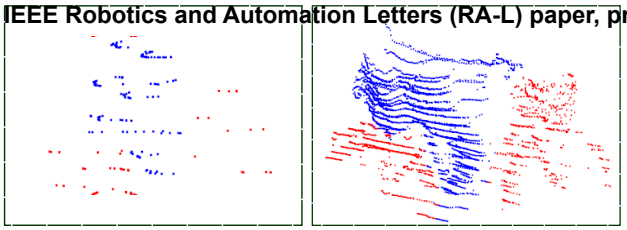

Fig. 4. *Left:* Point clouds of the motorcyclist far away from the low-resolution LiDAR, most of the data belong to the pedestrian. *Right:* Visualization of a detailed motorcyclist. Obviously, it is hard to recognize motorcycles as well as pedestrians.

Considering that the distance has only one dimension, we view the distance between dynamic objects and LiDAR as a word in NLP [40] and utilize the embedding function to convert the distance into high-dimensional vectors. Specifically, the distance is embedded to generate two forms of vectors, namely 128-dimensional and 256-dimensional vectors. The embedding function is usually exploited as the input layer of a neural network and its embedding vector can be used as a feature representation in subsequent tasks. In this way, it helps the model capture the connections between inputs, thereby improving the performance. In addition, the size of the dictionary is set to 200. The embedding parameters are also optimized during training. According to the experiments it is verified that using 128-dimensional vector to embed the distance obtains a higher classification accuracy. Therefore, we feed 896-dimensional vector as the input into the LSTM for each moment. The theoretical basis of this thought is that the difference of their speed may be used as one of the indicators of classification. Considering that the LSTM is a progressive classification network, it is believed that LSTM can learn the speed difference of dynamic objects according to the amount of distance change. In such a way, we improve the ability of classifying pedestrians and motorcycles.

Taking the distance into account, the specific formula of the fusion vector F is,

$$\mathrm{F} = [S_1, \cdots, S_{15}], \tag{1}$$

$$\mathrm{S}_i = \left[R_i^1 \bigoplus R_i^2 \bigoplus R_i^3 \bigoplus E\left(d_i\right) \right], \tag{2}$$

where S_i is the feature vector of the i^{th} scan, S_i concatenates R_i^j (the 256-dimensional vector of the j^{th} network) and $E(d_i)$ (returning the 128-dimensional embedding vector of the involved distance d_i). In the supplementary[1], we suggest incorporating the distance as the input rather than the output.

After the fusion of object features, two models are used to recognize time-series data. One directly uses multi-layer perceptron (MLP) to classify concatenated vector in 1.5 seconds, with 3 hidden layers and their sizes are 2048, 1024, and 256, respectively. Another exploits LSTM to divide feature vector into 15 moments as the input of the network, aiming to demonstrate its gradual classification ability to continuously improve the confidences of the results.

D. Loss Function

Since dynamic objects such as motorcyclists have point clouds that belong to pedestrian, motorcyclists and pedestrians share similar point cloud appearances, especially when the distance between the motorcyclist and the LiDAR is far away. Consequently, networks cannot distinguish motorcycle from pedestrian if the loss is calculated by multi-classification cross entropy. Therefore, this paper adopts the idea of multi-label classification and uses binary cross entropy instead of forcibly separating pedestrians and motorcyclists. Multi-label classification gives a set of labels for each target and the ranking within the set. In this example, the target output set should contain {motorcycle, pedestrian}. Otherwise, the output should be {pedestrian} or {car}. The loss function is,

$$\mathrm{Loss}(i,t) = -(t * \log(i) + (1-t) * \log(1-i)), \tag{3}$$

where i is the output vector of the network and t is the ground truth vector of dynamic objects. We use Sigmoid as nonlinear activation function to activate the last layer of the network. The underlying purpose is to minimize the difference between each label and the ground truth of a single object in the training set rather than focusing on the class with the highest score.

During the training, it is found that multi-labels of motorcyclists and pedestrians are more difficult to classify compared with cars. Therefore, we adopt Focal Loss [29] to assign different loss weights to separate categories according to the imbalance of samples and the difficulty of classification. In this way, the network is promoted to move toward the division of motorcyclists and pedestrians and improve the overall classification accuracy of dynamic objects. The loss is calculated according to,

$$\begin{aligned}\mathrm{Loss}(i,t) = -(&\alpha * (1-i)^{\gamma} * t * \log(i) \\ &+ (1-\alpha) * i^{\gamma} * (1-t) * \log(1-i)),\end{aligned} \tag{4}$$

where α is to control the imbalance of samples, γ enables the network to focus on difficult training samples.

IV. Experiment

A. Setup

Hardware Device. As shown in Fig. 2, we install RS-LiDAR-16[2] on the left and right side of an unmanned robot to collect data. It contains 16 scanning beams with a vertical resolution of 2.0° between adjacent beams. In the horizontal range, this LiDAR has a coverage of 360°, achieving real-time 3D imaging effect through rotation. Although RS-LiDAR-16 is a low-cost sensor, it still provides long detection distance of 150 meters and ensures the measurement accuracy of ± 2cm. For our experimental evaluation, experiments are performed on the computer set up with Ubuntu 20.04, Intel i7-7700HQ CPU (2.80GHz), 16-GB RAM and a NVIDIA GTX 1070 GPU.

Public Dataset. We select ModelNet40 [5] as the classification dataset for three point cloud deep learning networks. It contains 12,311 CAD model point clouds in 40 categories. This dataset only contains pedestrian and car, lacking motorcycle samples. In order to make the referee of the dynamic objects determine whether they have the characteristics of motorcycle point clouds before putting in LSTM, we join the motorcycle class of ShapeNet [31] into training, which contains 200

[2]https://www.robosense.cn/

TABLE II
PARAMETERS OF OUR PROPOSED DATASET

Scene	Time (s)	LiDAR (msgs)
Campus Scene 1	81	818
Campus Scene 2	107	1,078
Campus Scene 3	69	698
Campus Scene 4	274	2,748

bicycle, motorcycle, and electromobile samples. In our work, the base-classifiers are trained on 41 classes which contain 40 classes from ModelNet40 and one motorcycle class from ShapeNet. Considering the fact that it is hard to obtain the entire profile of a car during scanning, we cut each sample of a car into two parts which contain {side, front} and {side, rear} point clouds of the car. As for the domain adaption of the low-resolution point clouds, we set the number of point clouds used for training to 1,024, remaining the other parameters. The underlying reason is that we try to make the density of training samples as similar as possible to that of 16-beam point clouds without increasing the difficulty of model training.

Self-collected Dataset. As for the dataset of dynamic objects data from the low-resolution LiDAR, we utilize the above-mentioned unmanned robot to collect data in Tongji University which includes abundant cars, pedestrians and motorcycles. These dynamic objects are scanned at the frequency of 10Hz. We use the point cloud tool Semantic Segmentation Editor to manually annotate the dynamic objects as the ground truth to measure the accuracy of our PointLE. As for the data quality and diversity, we obtain dynamic object point clouds from various scanning directions (horizontal, vertical, and oblique) and different distances, which are manually annotated. In addition, we collect pedestrians of different heights, cars and motorcycles (e.g., motorcycles also include bicycles and electromobiles) of different types. Compared with existing benchmarks, we aim at low-resolution point clouds of dynamic objects. Table II shows the detailed information of our dataset, which covers 1,108 sequential car samples, 784 pedestrian samples and 750 motorcyclist samples after data fusion. Specifically, we split 1,531 sequences as the training set, 325 sequences as the validation set and 786 sequences as the test set. During the training process, Stochastic Gradient Descent is employed with a learning rate of 0.004 and a weight decay of 0.0001 in 200 epochs. In Focal Loss, we use α of 0.7 and γ of 0.6.

16-beam KITTI. We incorporate KITTI [6] as an external benchmark to further evaluate the capability of PointLE. The KITTI tracking dataset provides the annotations of object point clouds from 21 scenarios, which are collected by Velodyne HDL-64E. This LiDAR has 26.8 degrees of the vertical field of view and emits a total of 64 laser beams. Due to the different data densities between Velodyne HDL-64E and the 16-beam LiDAR, we obtain the 16-beam KITTI dataset through downsampling. By doing so, the resulting point clouds are with vertical resolutions of 2-degrees, as the same as the point clouds gathered from 16-beam LiDARs. After downsampling, based on the annotations provided by KITTI, we extract the point clouds of dynamic objects, including cars, pedestrians, and cyclists. Considering that our method utilizes sequence data as the input, we retain objects with at least 10 points per frame within 1.5 seconds and calculate their distances from the LiDAR. In this way, the point clouds and distances of dynamic objects are organized into sequences. Furthermore, we exploit the point cloud segmentation algorithm to remove ground point clouds of the sequences. As a result, we obtain a 16-beam KITTI dataset with 1080 car sequences, 1058 pedestrian sequences, and 318 cyclist sequences.

TABLE III
PERFORMANCE OF DIFFERENT CLASSIFICATION BASELINES ON OUR PROPOSED DATASET. BEST RESULTS IN BOLD. OA REFERS TO THE OVERALL ACCURACY.

	Car	Pedestrian	Motorcycle	OA
PointNet++ [20]	**99.79**	40.12	50.15	68.00
3DmFV [12]	94.39	84.73	83.93	88.66
DGCNN [16]	98.96	90.42	76.01	89.96
RSCNN [10]	**99.79**	80.24	83.28	89.30
PointASNL [22]	95.22	99.10	89.25	94.61
PointConv [11]	95.01	91.92	89.85	92.61
PF-NET [23] + PointNet++ [20]	89.40	-	82.39	-
PF-NET [23] + 3DmFV [12]	94.18	-	85.25	-
PF-NET [23] + DGCNN [16]	98.54	-	81.19	-
PointLE	98.59	**100.0**	**97.44**	**98.73**

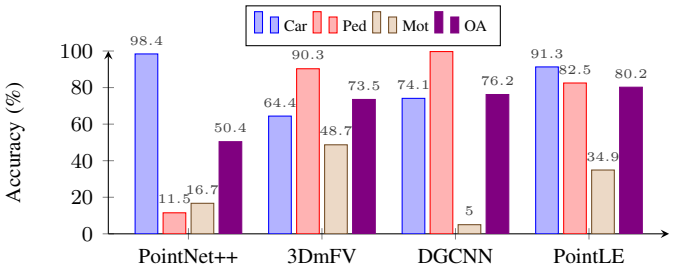

Fig. 5. Performance of different classification baselines and our PointLE on 16-beam KITTI dataset. Ped refers to the pedestrian, Mot to the motorcycle, and OA to the overall accuracy.

B. Experiment on Dynamic Object Classification

In Table III, we show the classification results of several representative point cloud deep learning networks and our PointLE. We pretrain these networks on the public dataset. Subsequently, they are fine-tuned during training on our collected dataset. Since our PointLE freeze the weights of these backbones to utilize their generalization abilities, these backbones are only fine-tuned in the output layer so as to make the comparison fair. As for the training detailes, Adaptive Moment Estimation (Adam) optimizer is employed with a learning rate of 0.001 in 100 epochs. The experiments indicate that different networks show their unique advantages of classifying separate categories of objects. By means of ensembling these networks, our PointLE achieves the outstanding classification accuracy of 98.73% over the competing state-of-the-art approaches by a large margin. From the perspective of single-scan point clouds completion, we utilize PF-Net [23] to interpolate point clouds of dynamic objects in a single scan before using other networks. According to the results in Table III, this method still cannot meet the classification requirements.

Furthermore we also conduct the classification experiment on 16-beam KITTI dataset. In order to evaluate the generalization of our PointLE, we utilize the model trained from the self-collected dataset to classify the objects in 16-beam KITTI. As shown in Fig. 5, our PointLE obtains notable improvements of overall accuracy compared with the SOTA counterparts. In fact, it is quite challenging to perform classification on 16-beam KITTI's point clouds with extreme sparsity. Considering

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2024, Yokohama, Japan. Cite as RA-L paper.

TABLE IV

ABLATION STUDY OF ENSEMBLE LEARNING RESULTS. WE MAKE A COMPARISON BETWEEN ONLY USING ONE BACKBONE ALONG WITH THE LSTM AND EXPLOITING THREE BACKBONES. POINTNET++, 3DMFV, AND DGCNN ARE SELECTED AS THE CANDIDATE BACKBONES. BEST RESULTS IN BOLD. OA REFERS TO THE OVERALL ACCURACY.

	Car	Pedestrian	Motorcycle	OA
PointNet++ [20] + LSTM	98.59	99.15	86.15	95.67
3DmFV [12] + LSTM	97.46	95.34	90.77	95.17
DGCNN [16] + LSTM	98.59	99.58	87.69	96.18
PointLE	**98.59**	**100.0**	**97.44**	**98.73**

TABLE V

COMPARISON OF TIME-SERIES FUSION RESULTS. MLP AND LSTM ARE SELECTED AS THE NETWORK STRUCTURES. THE EFFECTS OF DIFFERENT LOSS FUNCTIONS AND DISTANCES ARE TESTED. BEST RESULTS IN BOLD. OA REFERS TO THE OVERALL ACCURACY.

	Loss	Dist	Car	Pedestrian	Motorcycle	OA
MLP	Focal-Loss	/	98.31	99.15	81.54	94.40
MLP	Softmax-CE	✓	99.15	90.25	**97.95**	96.18
MLP	Binary-CE	✓	98.59	100.0	90.26	96.95
MLP	Focal-Loss	✓	98.31	100.0	84.62	95.42
LSTM	Focal-Loss	/	98.03	99.15	85.64	95.29
LSTM	Softmax-CE	✓	**99.44**	89.83	97.44	96.06
LSTM	Binary-CE	✓	98.59	99.58	90.26	96.82
LSTM	Focal-Loss	✓	98.59	**100.0**	97.44	**98.73**

that the horizontal resolution of Velodyne HDL-64E is one-third of that of a 16-beam LiDAR, the point clouds become much sparser after downsampling (e.g., some dynamic objects have only 10 points in total consequently). Moreover, there are many occlusion situations in 16-beam KITTI, which undoubtedly also increases the difficulty of classification. As for the scenario of data collection, the point clouds in 16-beam KITTI are collected in a high speed, which brings more variations in the scanned data. Nevertheless, the superior generalization of our PointLE is verified through evaluations on the challenging 16-beam KITTI without any additional training.

C. Ablation Study on Ensemble Learning

We conduct the ablation study to demonstrate the effect of ensemble learning. Specifically, the comparison of only using one backbone along with the LSTM and exploiting three backbones is added in this section. As shown in Table III and Table IV, we utilize time-series fusion to improve the classification accuracy of dynamic objects, especially on pedestrians and motorcycles. Even if we only use one backbone to encode the object feature, it is able to indicate change rules of object representation during movements. Moreover, the idea of ensemble learning fuses different representation abilities of three backbones to enhance the object feature, improving the accuracy to 98.73%.

D. Experiment on Time-series Fusion

Fusion Model. In time-series fusion classification, we conduct experiments based MLP and LSTM. When resorting to MLP, by adjusting the number of hidden layers and the number of neurons in each hidden layer, we find out that the multi-layer perceptron with three hidden layers of 2048, 1024, and 256 obtains the best classification result. According to Table V the overall classification accuracy achieves 96.95%. As for LSTM, we design three forms of 1.5 seconds, namely 15, 5, and 3 input moments and the former one is verified to be the optimum. In the network structure, we exploit the hidden vector size of 512 and the embedding dimension of distances is selected as 128, 256, or 512. Furthermore, we improve generalization by setting up Dropout of 0.2. The experiment shows that embedding the distance to 128 dimensions obtains a best overall classification accuracy of 98.73%. It well verifies that LSTM shows its advantage of gradual classification.

Loss Function. Considering that the classification difficulty of different objects varies, for example, distant motorcyclists are difficult to divide with pedestrians due to the small proportion of bicycle point clouds. To cope with it, we adopt Focal Loss [29] instead of softmax and binary cross entropy in the training process. According to Table V when Focal Loss [29] is exploited, the network focuses more on how to identify pedestrians and motorcyclists. Although the classification accuracy of the car drops slightly, the classification results improve significantly for both pedestrians and motorcyclists.

Distance of Dynamic Objects. As shown in Table V, We also consider the effect of adding the distance between objects and the LiDAR on the classification accuracy. The distance is fed into LSTM along with three high-dimensional features as the input, improving the classification accuracy of motorcyclists to 98.73%. Considering that LSTM is a gradual classification network, it can learn the speed difference of objects according to the change of distances.

E. Analysis on Generalization

After removing ground points, single objects are generated from the ground truth. Indeed, segmenting and tracking sparse point clouds are also difficult tasks. In the common scenarios, there exist various dynamic objects which are relatively easy or hard to detect. As for the easy ones, they are usually isolated objects which can be successfully segmented and tracked. Nevertheless, there are occasional cases of segmenting and tracking failures caused by adjacent or high-distance objects. In order to evaluate the generalization of PointLE in those cases, we simulate difficult samples for experiments.

Segmenting. Through data simulation, we test the impact of the under-segmentation and over-segmentation targets on classification. Specifically, we randomly discard point clouds or add noisy points according to a certain proportion. We set four segmentation error ratios of 1%, 2%, 5%, and 10%. It is worth mentioning that we do not add random noise within the range of the bounding box of the object. Considering that the adjacent point clouds of objects tend to interfere with the segmentation results in the actual scene, we randomly select a certain proportion of the current point clouds and add noise within 5cm. Subsequently, we classify the target point cloud with simulated segmentation as the input of PointLE.

Tracking. Sometimes there is occlusion, which leads to the failure of tracking the target. We simulate the tracking sequence of each dynamic object under different occlusion duration. Specifically, other dynamic object data that are close to the object are regarded as occlusions and inserted into the tracking sequence. We set the occlusion duration to 2∼6 scans.

According to Fig. 6, the precision of our PointLE is interfered when taking simulated segmenting and tracking into account. When the 10% error ratios of segmenting and tracking simulation are added at the same time, the accuracy of the

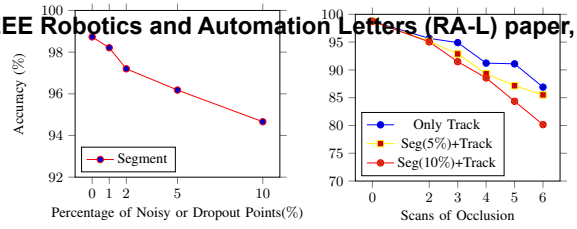

Fig. 6. Classification results of simulated segmenting and tracking point clouds. The former one means adding noisy points or discarding points by different percentages. The latter one inserts multiple scans of other dynamic objects into the sequences as occlusion. Even if we consider segmenting and tracking errors, the accuracy decreases slightly and still reaches 80.15%.

classifier decreases to 80.15%. Therefore, it is verified that the generalization of PointLE is able to meet the requirement for dynamic object classification.

V. Conclusion

In this paper, we present a new approach to classify dynamic objects data from low-resolution LiDAR. Due to the different representation abilities of various networks, our method proposes a point cloud fusion strategy based on ensemble learning. Afterwards, we utilize LSTM to achieve gradual classification in a time series. A dataset of dynamic objects is also provided to evaluate our approach. Experimental results on the this dataset illustrate the effectiveness of our method.

References

[1] F. Yang, Z. Zhu, X. Gong and J. Liu, "Real-time dynamic obstacle detection and tracking using 3D lidar," *J. Zhejiang Univ., Eng. Sci.*, vol. 46, no. 9, pp. 1565–1571, 2012.

[2] S. Zhang, "Dynamic obstacle recognition and tracking in sparse lidar point cloud for low-speed unmanned driving," M.S. thesis, Dept. Surveying Geo-Informatics, Tongji Univ., Shanghai, China, 2021.

[3] A. Teichman, J. Levinson and S. Thrun, "Towards 3D object recognition via classification of arbitrary object tracks," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2011, pp. 4034–4041.

[4] J. Wu, H. Xu, Y. Zheng, Y. Zhang, B. Lv and Z. Tian, "Automatic vehicle classification using roadside lidar data," *Transp. Res. Rec.*, vol. 2673, no. 6, pp. 153–164, 2019.

[5] Z. Wu *et al*, "3D ShapeNets: A deep representation for volumetric shapes," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2015, pp. 1912–1920.

[6] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? The KITTI vision benchmark suite," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2012, pp. 3354–3361.

[7] D. Maturana and S. Scherer, "VoxNet: A 3D convolutional neural network for real-time object recognition," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2015, pp. 922–928.

[8] Y. Li, R. Bu, M. Sun, W. Wu, X. Di and B. Chen, "PointCNN: Convolution on x-transformed points," in *Proc. Adv. Neural Inf. Process. Syst.*, 2018, pp. 828–838.

[9] C. R. Qi, H. Su, M. Nießner, A. Dai, M. Yan and L. J. Guibas, "Volumetric and multi-view CNNs for object classification on 3D data," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 5648–5656.

[10] Y. Liu, B. Fan, S. Xiang and C. Pan, "Relation-shape convolutional neural network for point cloud analysis," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 8887–8896.

[11] W. Wu, Z. Qi and L. Fuxin, "PointConv: Deep convolutional networks on 3D point clouds," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 9613–9622.

[12] Y. Ben-Shabat, M. Lindenbaum and A. Fischer, "3D point cloud classification and segmentation using 3D modified fisher vector representation for convolutional neural networks," *IEEE Robot. Automat. Lett.*, vol. 3, no. 4, pp. 3145–3152, 2018.

[13] S. Kumawat and S. Raman, "LP-3DCNN: Unveiling local phase in 3D convolutional neural networks," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 4898–4907.

[14] W. Shi and R. Rajkumar, "Point-GNN: Graph neural network for 3D object detection in a point cloud," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 1708–1716.

[15] H. Su, S. Maji, E. Kalogerakis and E. Learned-Miller, "Multi-view convolutional neural networks for 3D shape recognition," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2015, pp. 945–953.

[16] Y. Wang, Y. Sun, Z. Liu, S. E. Sarma, M. M. Bronstein and J. M. Solomon, "Dynamic graph CNN for learning on point clouds," *ACM Trans. Graph.*, vol. 38, no. 5, pp. 1–12, 2019.

[17] A. Cheraghian and L. Petersson, "3DCapsule: Extending the capsule architecture to classify 3D point clouds," in *Proc. IEEE Winter Conf. Appl. Comput. Vis.*, 2019, pp. 1194–1202.

[18] J. Li, B. M. Chen and G. H. Lee, "SO-Net: Self-Organizing network for point cloud analysis," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 9397–9406.

[19] R. Q. Charles, H. Su, M. Kaichun and L. J. Guibas, "PointNet: Deep learning on point sets for 3D classification and segmentation," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 77–85.

[20] R. Q. Charles, L.Yi, H.su, L. J. Guibas, "PointNet++: Deep hierarchical feature learning on point sets in a metric space," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 5105–5114.

[21] H. Su *et al*, "SPLATNet: Sparse lattice networks for point cloud processing," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 2530–2539.

[22] X. Yan, C. Zheng, Z. Li, S. Wang and S. Cui, "PointASNL: Robust point clouds processing using nonlocal neural networks with adaptive sampling," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 5588–5597.

[23] Z. Huang, Y. Yu, J. Xu, F. Ni and X. Le, "PF-Net: Point fractal network for 3D point cloud completion," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 7659–7667.

[24] W. Yuan, T. Khot, D. Held, C. Mertz and M. Hebert, "PCN: Point completion network," in *Proc. Int. Conf. 3D Vis.*, 2018, pp. 728–737.

[25] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, 1997.

[26] A. Vaswani *et al*, "Attention is all you need," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 6000–6010.

[27] M. Himmelsbach, F. v. Hundelshausen and H. -J. Wuensche, "Fast segmentation of 3D point clouds for ground vehicles," in *Proc. IEEE Intell. Vehicles Symp.*, 2010, pp. 560–565.

[28] D. Zermas, I. Izzat and N. Papanikolopoulos, "Fast segmentation of 3D point clouds: A paradigm on lidar data for autonomous vehicle applications," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2017, pp. 5067–5073.

[29] T. -Y. Lin, P. Goyal, R. Girshick, K. He and P. Dollár, "Focal loss for dense object detection," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2017, pp. 2999–3007

[30] Y. Li *et al*, "A scalable active framework for region annotation in 3D shape collections," *ACM Trans. Graph.*, vol. 35, no. 6, pp. 1–12, 2016.

[31] H. Fan and Y. Yang, "PointRNN: Point recurrent neural network for moving point cloud processing," 2019.

[32] R. Huang, W. Zhang, A. Kundu, C. Pantofaru, D. A. Ross, T. Funkhouser, and A. Fathi, "An LSTM approach to temporal 3D object detection in LiDAR point clouds," in *Proc. Eur. Conf. Comput. Vis.*, 2020, pp. 266–282.

[33] J. Yin, J. Shen, X. Gao, D. J. Crandall and R. Yang, "Graph neural network and spatiotemporal transformer attention for 3D video object detection from point clouds," in *IEEE Trans. Pattern Anal. Mach. Intell.*, 2021, pp. 1–1.

[34] J. Yin *et al*, "ProposalContrast: Unsupervised pre-training for LiDAR-based 3D object detection," in *Proc. Eur. Conf. Comput. Vis.*, 2022, pp. 17–33.

[35] X. Li, J. Yin, B. Shi, Y. Li, R. Yang and J. Shen, "LWSIS: LiDAR-guided weakly supervised instance segmentation for autonomous driving," in *Proc. AAAI Conf. Artif. Intell.*, 2023.

[36] Y. Wang, J. Yin, P. Frossard, R. Yang and J. Shen, "SSDA3D: Semi-supervised domain adaptation for 3D object detection from point cloud," in *Proc. AAAI Conf. Artif. Intell.*, 2022.

[37] J. Yin *et al*, "Semi-supervised 3D object detection with proficient teachers," in *Proc. Eur. Conf. Comput. Vis.*, 2022, pp. 727–743.

[38] J. Yang, S. Shi, Z. Wang, H. Li and X. Qi, "ST3D: Self-training for unsupervised domain adaptation on 3D object detection," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2021, pp. 10363–10373.

[39] J. Yuan *et al*, "Bi3D: Bi-domain active learning for cross-domain 3D object detection," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2023.

[40] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," in *Proc. Int. Conf. Learn. Represent.*, 2013.