Deep Instance Segmentation with High-Resolution Automotive Radar

Jianan Liu†, Weiyi Xiong‡, Liping Bai‡, Yuxuan Xia§, Bing Zhu¶∗

Abstract—Automotive radar provides reliable environmental perception in all-weather conditions with affordable cost, but it hardly supplies semantic and geometry information due to the sparsity of radar detection points. With the development of high-resolution automotive radar in recent years, instance segmentation becomes possible by using automotive radar. Its data contain rich contexts such as Radar Cross Section and micro-Doppler effects, and sometimes can provide detection when the field of view is obscured. The outcome from instance segmentation could be potentially used as the input of trackers for tracking targets. In this paper, we propose two efficient methods for instance segmentation with radar detection points, one is implemented in an end-to-end deep learning driven fashion using PointNet++ framework, and the other is based on clustering of the radar detection points with semantic information. Both approaches can be further improved by implementing visual multi-layer perceptron. The effectiveness of the proposed methods is verified using experimental results on the recent RadarScenes dataset.

Index Terms—Autonomous driving, environmental perception, instance segmentation, semantic segmentation, clustering, classification, automotive radar, deep learning

I. INTRODUCTION

In the field of autonomous driving, automotive radar plays an important role in environmental perception due to its affordable costs, inherent measurement of object relative velocity, and reliability in all-weather conditions, as compared to camera and Lidar[1][2]. The data representation of an automotive radar is usually a set of detection points (or a point cloud), which are generated by pre-processed raw radar data, typically in the form of a range-Doppler map or a range-azimuth heatmap. Compared to Lidar points, radar points usually provide more information, e.g. velocity (Doppler) and the radar cross section (RCS) values. In addition, radar points are much sparser than Lidar points due to the low resolution of radar, resulting in a lack of semantic and geometric information. Thus, it is unsuitable to directly apply methods developed for Lidar points to radar points.

A common way to process point clouds is to either transform point clouds into 3D grid-like representation called voxel or 2D grid-like representation in the bird eye’s view (BEV), or project them into range view [3], and then use Convolution Neural Network (CNN) to perform classification, detection or segmentation. The main problems of above methods are the high computational and memory cost, and the quantization error introduced during the point-to-voxel/pixel transformation. In addition, early stages of CNN has difficulty in capturing the spatial interactions of radar points due to their sparsity, and thus such methods may be inapplicable for radar point cloud.

Another common way is to use raw points as input, regarding spatial coordinates as part of the features and putting them in the channel dimension [4][7]. This method is more efficient and could overcome the other problems that the former confronts with. Typical examples of the second method include the PointNet [4][5] and its variants, and in this work we choose to use PointNet++ [5], as the backbone of our proposed network.

The main contributions of this paper include:

• Two new strategies for automotive radar-based instance segmentation are proposed. One is end-to-end instance segmentation with a modified loss function of Similarity Group Proposal Network (SGPN) [6]; the other is semantic segmentation based clustering, and we add a “center shift vector (CSV) prediction” branch to the semantic segmentation version of PointNet++ [5] to improve the performance.

• Both strategies are enhanced by incorporating visual Multi-Layer Perceptron (MLP) [8][10], facilitating the perceptional capability of global information of radar detection points in each frame.

• Experimental results on the recent RadarScenes dataset [11] show that the proposed strategies outperform the baseline (i.e. clustering based classification) by a large margin. Specifically, our selected method attains 88.5% mCov (mean coverage) and 85.2% mAP0.5 (mean average precision), which is 9.0% and 9.2% higher than the baseline, respectively.

The rest of this paper is organized as follows. Related works of clustering methods, including visual MLPs, radar-based perception, and neural networks processing point cloud data are introduced in Section II. The proposed two radar-based instance segmentation strategies and their enhancements are described in detail in Section III. Experimental results on the RadarScenes dataset are provided and discussed in Section IV. Some concluding remarks and future work suggestions are given in the final section.
II. RELATED WORKS

A. DBSCAN and its Modification

As mentioned earlier, the generic method of radar-based instance segmentation is clustering based classification. Among different clustering methods, DBSCAN (Density-Based Spatial Clustering of Applications with Noise) [12] algorithm is more suitable for radar detection points. To get better performance, [13] proposed REDBSCAN (Radar Elliptical Density-Based Spatial Clustering of Applications with Noise) by leveraging the radar resolution in clustering. However, this method only uses the spatial coordinates. To make fully use of all features of a point, other modified DBSCAN algorithms are proposed. [14] take velocity and the received power into consideration and change the distance metric by using certain footprint. Although not for radar points exclusively, ST-DBSCAN [15] is also devised to process spatial-temporal data, where the time information of points can be seen as a non-spatial feature.

B. Point Cloud Processing

Point clouds are sporadic and permutation invariant, making effective information extraction challenging. While the image processing techniques, such as 2D convolution, can be extrapolated into the realm of 3D point cloud data processing, the outcomes of such approaches turn out to be ineffective.

PointNet [4] and the subsequent variants [5] are network structures designed specifically for point cloud data, where the input data points are projected into a higher dimension space before going through a permutation invariant function, e.g., a max pooling function, for feature extraction. For classification tasks, the output of the max pooling function is utilized for the final prediction, whereas for segmentation tasks, the output of the max pooling function is then combined with other information during the feature propagation layers, and the network also outputs the classification of each data point.

As opposed to taking all the data points as the input of the first layer, as does PointNet, PointNet++ seeks to imitate the convolution layer of 2D images, in an attempt to capture the local context. To achieve this, a Set Abstraction (SA) layer is used to sample, group points and capture local structure. Furthermore, the Feature Propagation (FP) layer is devised to propagate features from sampled points to original points and get point-wise features, for the purpose of segmentation.

PointNet and PointNet++ do not provide the function of direct instance segmentation, but some efforts have been made toward this direction. For instance, SGPN [6] takes PointNets as its backbone and introduce a similarity matrix for instance segmentation. And HAIS [7], an approach which combines PointNets with clustering, adopts hierarchical aggregation to progressively generate instance proposals, so as to overcome traditional clustering problems.

C. Visual MLPs

Attention-based transformers [16–18] are popular approaches for computer vision tasks, but some recent works prove that network based on only MLPs achieve comparable performance [10]. MLP-mixer [9] replaces the multi-head self attention [19] with a linear layer implemented on the spatial dimension. To make the MLP flexibility to receive images of different sizes, the authors in [20] propose cycle-MLP, a cycle Fully Connected (FC) layer and use it to change the spatial MLP in MLP-mixer. However, as the data structure of images and point clouds are different, such method cannot be applied to our work without voxelizing the point cloud. gMLP [10] discards the multi-head self attention in transformer and add a spatial gating unit to capture spatial interaction. Taking the inspiration from self-attention, external attention [8] uses two linear layers with a double normalization in between, to reduce the computational complexity.

D. Automotive Radar-based Perception

Automotive radar-based perception, including semantic segmentation, clustering, classification, instance segmentation, object detection and tracking, has played an important role in the modern ADAS and autonomous driving system. With the availability of large-scale radar datasets [11], automotive radar detection points based perception has been investigated recently. A two stage clustering algorithms is designed in [21] and estimated state information by using extended target tracking algorithm is employed as prior information to provide more stable clustering in [22]. Machine learning and deep learning methods have also been explored for automotive radar detection points based perception. For example, the radar detection points are used as input data for semantic segmentation task [23], together with occupancy grid representation of environments [24] [25]. Such detection points representation of radar data is also employed for classification, object detection and tracking purpose. For example, [26–28] utilize random forest and LSTM as classifier to estimate the type of clustered detection points, [29] modifies the PointNets for object detection but only one class (cars) are considered, and [30] performs detection and tracking by adopting combinations of PointNet++ based neural network with Kalman filter and global nearest neighbor for ID assignment over multiple frames.

On the other hand, compared to radar detection points based perception, perception can be performed by leveraging the richer feature information contained in the raw radar data like range-azimuth heatmap [31]. In addition, [32] segments the radar data semantically in multi-view representations of range-Doppler-azimuth 3D cube.

III. PROPOSED METHODS

The most commonly used framework for automotive radar-based perception is two-stage based, where the first stage is to extract the measured state information of each object by either clustering or instance segmentation on radar detection points, and the second stage is to input the measured information to the tracking filter to refine the localization information and provision of ID for each object. The performance of the first stage (i.e., clustering or instance segmentation) is the bottleneck of the entire perception pipeline, since traditional clustering algorithm (e.g., DBSCAN) can only process
the geometry information. Specifically, without carefully selected hyper-parameters, DBSCAN is capable of clustering points from multiple small and closely-spaced targets into one instance, or clustering points from a large target into several separate instances. To tackle such problem, we propose two deep-learning based strategies for automotive radar instance segmentation by directly processing the detection points, namely end-to-end instance segmentation and semantic segmentation based clustering. The two proposed strategies are based on radar detection points instead of raw level radar data (e.g., the range-Doppler map or range-azimuth heatmap), and these strategies are applicable to any automotive radar-based perception pipeline, because radar detection points can be accessible from any automotive radar.

In this section, the widely accepted “clustering first classification later” strategy for automotive radar-based instance segmentation is first introduced as the baseline. The proposed end-to-end instance segmentation and semantic segmentation based clustering strategies are then described in detail. The proposed methods are enhanced with MLPs to improve the overall performance.

A. Baseline Method

The commonly-used method for radar-based instance segmentation is clustering based classification [27, 28, 33]. As illustrated in Fig. 1, the input point cloud is first clustered, and then each cluster is sent to a classifier (e.g., support vector machine or random forest classifier) to get the predicted class.

In this paper, DBSCAN is chosen as the baseline method for clustering, and random forest classifier is used to predict scores for all classes. The hand-crafted features of every cluster estimated from DBSCAN are used as input of random forest classifier, including the mean values and deviations of range, azimuth, Doppler and RCS.

B. End-to-End Instance Segmentation

To implement instance segmentation in a deep learning framework, the proposed model is built upon SGPN [6], as illustrated by Fig. 2. This model connects three heads to PointNet++, and predicts a similarity matrix $S$ to estimate the possibility that any pair of the points belong to the same instance, a similarity confidence map $M_{CF}$ to estimate the uncertainty of similarity results, and a semantic segmentation map $M_{SEM}$ to provide the semantic information estimation of every point.

The loss functions of $S$ and $M_{SEM}$ remain as double hinge loss and cross entropy loss. However, the binary cross entropy loss (BCE), instead of the mean square error (MSE, or $l_2$), is set as the loss function of $M_{CF}$ to facilitate uncertainty modelling, i.e.,

$$L_{CF} = -\frac{1}{N} \sum_{i=1}^{N} [\text{IoU}(g_i, p_i) \times \log(M_{CF,i}) + (1 - \text{IoU}(g_i, p_i)) \times \log(1 - M_{CF,i})]$$

where $N$ is the number of the predicted group (or equivalently, the number of points) in a frame; $M_{CF,i}$ is the $i$-th row of confidence map; and $\text{IoU}(g_i, p_i)$ is the Intersection over Union (IoU) between the $i$-th predicted group and the corresponding ground truth group. Specifically,

$$\text{IoU}(g, p) = \frac{|g \cap p|}{|g \cup p|}$$

where $g$ denotes the set of points belonging to a ground truth instance, $p$ is the set of points in a predicted group, and $|\cdot|$ denotes the set cardinality (i.e., the number of elements in the set).

C. Semantic Segmentation based Clustering

Another way to implement instance segmentation is to reverse the operation order of the baseline, i.e., semantic segmentation (point-wise classification) based clustering. Therefore, different clustering parameter settings are used for different classes. Specifically, let the network model concentrate on semantic segmentation, and then apply the clustering for each group of detection points with the same semantic information.

Remark 1: It is intuitive that clustering points with semantic information may achieve better performance, since attributes of clusters belong to various class types might be significant different. For instance, there may be more than 10 detection points from a large vehicle, and the distances between these detection points could be larger than those from a two-wheeler, while a pedestrian may only have one detection point.

As standalone PointNet++ based semantic segmentation cannot provide satisfactory results for radar detection points due to the sparsity of radar detection points, we introduce a CSVs prediction branch in our semantic segmentation network to estimate the offset between every point and the geometric center of corresponding ground-truth instance. In this way, the detection points are assembled by shifting the
center of the instance to facilitate clustering. Since the detection points carry no systematic RCS information, we carefully tailor the CSVs prediction branch such that only coordinate and velocity are used to predict the CSV for each radar detection point. The proposed architectures of the prediction heads are two-layer MLPs, as illustrated in Fig. [3]

The total loss is obtained by

$$L = L_{SEM} + \alpha L_{SHIFT} = L_{SEM} + \alpha (L_{CS} + L_{NIP})$$  \hspace{1cm} (3)

where $\alpha > 0$ is a weighting factor and $L_{SEM}$ is the cross entropy loss for semantic segmentation head. Here, the loss for prediction of CSVs is defined by $L_{SHIFT} = L_{CS} + L_{NIP}$, where $L_{CS}$ and $L_{NIP}$ are the Cosine Similarity (CS) loss and Normalized Inner Product (NIP) loss, respectively, calculated by

$$L_{CS} = [1 - \cosine_similarity(\Delta x_{\text{pred}}, \Delta x_{\text{gt}})]$$ \hspace{1cm} (4)

$$L_{NIP} = \left| \frac{\text{inner_product}(\Delta x_{\text{pred}}, \Delta x_{\text{gt}})}{\|\Delta x_{\text{gt}}\|} - 1 \right|$$ \hspace{1cm} (5)

where $\epsilon$ is a small positive number (e.g., $10^{-5}$) to prevent singularity; $\Delta x_{\text{pred}}$ denotes the predicted CSV between every detection point; $\Delta x_{\text{gt}}$ denotes the geometric center of corresponding instance and its ground truth value; $\cosine_similarity(\cdot, \cdot)$ and $\text{inner_product}(\cdot, \cdot)$ calculates the included angle and the inner product between two feature vectors, respectively. Specifically, if $x_1$ and $x_2$ are vectors with the same dimension, then

$$\cosine_similarity(x_1, x_2) = \frac{x_1^T x_2}{\|x_1\| \|x_2\|}.$$ 

$$\text{inner_product}(x_1, x_2) = x_1^T x_2.$$ 

**Remark 2:** Different from $l_2$ loss in [7], the above definition of $L_{SHIFT}$ fully explores the offset shifting generated from feature vectors in the latent space by leveraging the multi-dimension physical feature information from the radar.

**Remark 3:** As shown in (4) and (5), the included angle approaches its ground truth due to the CS loss; and the NIP loss is designed for length approximation. Compared to $l_2$ loss, the proposed loss function for CSVs leads to a significant improvement in semantic segmentation of radar detection points, which can be seen in Section [IV]

During the inference time, predicted CSVs are used to push detection points towards the center of the instances. Since detection points assigned with different semantic information scarcely belong to the same instance, several clustering approaches are implemented with different parameters in parallel to estimate the final instances. Each clustering approach is for one semantic group, and it gathers all shifted detection points with the same estimated semantic information. The entire procedure is illustrated by Fig. [4]

### D. Enhancement with Visual MLPs

Attention-based visual transformers for image processing and perception have been extensively studied in recent years, and some of them focus on point cloud datasets. For example, 10 different attentions/transformers were applied in [34] to point cloud data, and comparisons are provided among them with respect to their structures and performances. However, few visual MLPs have been used for point cloud data processing. Compared to transformer architectures, the modules in visual MLPs are consistent with the PointNet++ MLP framework, facilitating the combination of these two modules. In this paper, visual MLPs are integrated into the proposed algorithm through an MLP block after each SA and FP layer in PointNet++. The extracted feature vectors are fed into an MLP block for down-sized feature refinement and propagated into next layer in the encoder of the network. In the decoder, the up-sampled feature vectors are strengthened by an MLP block to achieve better representations in the latent space. The structure of the enhanced network is shown in Fig. [5]

The MLP block in Fig. [5] is used to propagate multiple feature vectors, and it can be any visual MLP, e.g., MLP-Mixer [9], external attention [8], or gMLP [10]. In this research, the gMLP (MLP with gating units) is adopted. Note that in radar detection point representations, instance and semantic information of different objects can potentially be represented by the points-wise information from the same instance. For example, the size of the instance could be represented by the number of reflected points from the object, and a particular attribute (channel) information from the group of points belonging to the same object could be similar (e.g., the points from the same instance probably carry similar velocity and RCS). In addition, the position differences between points
Conclusion and Further Work

Further Work:

RCS the followings are useful:

in both Cartesian and polar coordinates. Among there data, (FOV) can be seen in Fig. 6. The illustration of radars in radarscenes dataset [11].

The radar provides data of position, velocity, time, and ID, max

of points (for convenience, the number of points in

point clouds from different driving environments, which are manually annotated with a class, an instance ID and other information. Specifically, there are more than 4 hours recording in this dataset, and it is composed of 158 sequences. In order to reduce the difference among training set, validation set and testing set, we shuffle and split the frames by the proportion of 8:1:1.

A. Dataset

RadarScenes [11] dataset is selected to validate the proposed methods. The dataset contains data from four front-mounted near-range radars, one camera, and one odometer. The four radars are 77 GHz near field automotive radar with a detection range of up to 100 meters. Each radar covers a ± 60° field of view. The four radars are mounted at the front end of the vehicle at 85°, 25°, −25°, and −85° with respect to the driver, respectively. The data stream is timestamped so that the ego-coordinate of any vehicle can be used as the anchor coordinate system, and information from all four radars are synchronized in one frame. The average frame rate is 17 Hz. Each frame contains points ranging from 28 to over 1000 (on average there are 158 points per frame). The illustration of mounting positions of four radars and the corresponding field of view (FOV) can be seen in Fig. 6.

The radar provides data of position, velocity, time, and ID, in both Cartesian and polar coordinates. Among these data, the followings are useful: RCS in dBsm; vr(compensated) in m/s, which is the radial velocity for this detection but compensated for the ego-motion; x_\text{cc} in meters, which is the position of the detection horizontal to the car in the car coordinate system (origin is at the center of the rear-axle); y_\text{cc} in meters, which is the position of the detection orthogonal to the car in the car coordinate system (origin is at the center of the rear-axle).

In general, RadarScenes contains real-world radar detection point clouds from different driving environments, which are manually annotated with a class, an instance ID and other information. Specifically, there are more than 4 hours recording in this dataset, and it is composed of 158 sequences. In order to reduce the difference among training set, validation set and testing set, we shuffle and split the frames by the proportion of 8:1:1.

B. Settings

Parameters of all experiments are set to the same values: the batch size is 512, the initial learning rate is 10^{-5}, and the optimizer is Adam. The learning rate restarts every 5 epochs with the scheduler of Cosine Annealing Warm Restarts.

Although there are 12 classes of objects in RadarScenes, we choose the settings of 5 classes, including settings of car, pedestrian, group of pedestrians, large vehicles and bicycles. The settings of the 5 classes merge into one, due to lack of data in some classes. In addition, static points are not used, since the overwhelming majority of the detection points in the dataset is static. It will be sufficiently accurate if the model learns an all-static prediction.

To solve the problem that every frame has different number of points (for convenience, the number of points in i-th frame is denoted by N_i), points are sampled randomly in each frame. Some statistics are obtained such as \max(N_i) and \text{mean}(N_i) to determine the sample size. In training, the sample size should be larger than most N_i because N_i is usually small, but it is unnecessary to be larger than \max(N_i) as it would increase the computational cost. However, while inferring, the sample size must be larger than \max(N_i), otherwise some detection points will be lost. In practice, the number of non-static points in a frame varies from 1 to 173, and less than

IV. EXPERIMENTS AND RESULTS

From the same instance indicate the size of the instance and its type. Since both random order and random reproduction of detection points have been applied for each epoch where massive number of combinations of points-wise interactions could be provided, features can be extracted from points-wise interactions globally, although the permutation invariant seems hardly to be hold anymore by performing the 1D convolutional operation in MLP-Mixer along points dimension. Furthermore, gMLP is an improved version of MLP-Mixer, and it combines the spatial gating mechanism with MLP-Mixer. Through this, it may provide more flexibility to tune the feature vectors, thereby improving the interaction between the number of detection points and features per point. The performance of the network with different visual MLPs will be compared and analysed in Section IV.

The structure of the MLP based PointNet++ network. The MLP block can be gMLP, MLP-Mixer, external attention and other MLP blocks that can take in multiple feature vectors.

Fig. 5. The structure of the MLP based PointNet++ network. The MLP block can be gMLP, MLP-Mixer, external attention and other MLP blocks that can take in multiple feature vectors.

Fig. 6. The illustration of radars in radarscenes dataset [11].
0.4% of frames have more than 100 points, so the sample size is set to 100 in training and 200 in testing, except the gMLP based network, whose parameters contains the sample size, and 200 sample size is set in both training and testing.

By sampling (or repeating), there exist 100 non-static points in each frame. We approximate the FOV to a $100m \times 100m$ field, taking into account the configuration and technical specification of the four near field radars. Based on these numerical structures, we require that the PointNet++ segmentation network \cite{qi2017pointnet++} have two SA Layers and two corresponding FP Layers. The number of sampled points in the first SA Layer is set to 64 with radius 8m. These parameters are designed such that all the sampling cycles would cover the entire FOV with appropriate overlapping:

$$n \pi r^2 > S_{FOV}$$

where $n$ is the number of sampled points; $r$ denotes the radius; and $S_{FOV}$ is the area of FOV.

The density of data points at each frame is $\frac{300}{72} = 3.125$; therefore, there are on average 3.1 data points in each sampling cycle. The sampling number is set to 8 which is larger since the maximum pooling operation of the PointNet++ network is duplication insensitive and we want to guarantee no undersampling. This design logic is extended to the second SA Layer, only with the input data being reduced to 64.

In practice, the same network structure is applied to both instance segmentation and semantic segmentation:

$$SA(64, 8, [8, 32, 64])$$
$$SA(16, 16, [64, 128, 256])$$
$$FP(64, 32)$$
$$FP(32, 32, 16)$$

where the notations are the same as those in PointNet++ \cite{qi2017pointnet++}. Specifically, $SA(K, r, [l_1, ..., l_d])$ means a set abstraction level sampling $K$ points, and the searching radius for each sampled point is $r$, which is followed by a PointNet of $d \times 1 \times 1$ convolution layers whose output channels are $l_1, ..., l_d$, respectively; $FP(l_1, ..., l_d)$ is a feature propagation level with $d \times 1 \times 1$ convolution layers. The final output of PointNet++ backbone is denoted as $F$. Note that for the enhanced models, the visual MLP block does not change the dimension of output feature vectors thus it could be appended in following of each $SA$ or $FP$ directly.

The structure of heads for instance segmentation are shown below:

$$F_{SIM} = \text{conv}(F, 16), S_{ij} = \|F_i - F_j\|_2$$
$$F_{CF} = \text{conv}(F, 16), M_{CF} = \text{conv}(F_{CF}, 1)$$
$$F_{SEM} = \text{conv}(F, 16), M_{SEM} = \text{conv}(F_{SEM}, n_{class})$$

where $\text{conv}(X, l)$ denotes a $1 \times 1$ convolution layer with input $X$ and output channel $l$; $F_k$ is the feature vector of the $k$-th point; $S_{ij}$ is the similarity value in position $(i, j)$ of $S$; and $n_{class}$ is the number of classes, which is 5 in our experiment. BatchNorm, ReLU and Dropout are used between two consecutive convolution layers.

### Table I: The four relationships between the prediction and ground truth in classification tasks

| Relationships | Predicted Class = Actual Class? |
|--------------|---------------------------------|
| Actual Class | True                            |
| Other Classes| False                           |
| True Positive (TP) | False Positive (FP) | True Negative (TN) |

Structures of heads for semantic segmentation are as the follows:

$$F_{SEM} = \text{conv}(F, 16), M_{SEM} = \text{conv}(F_{SEM}, n_{class})$$
$$F_{SHIFT} = \text{conv}(F, 16), M_{SHIFT} = \text{conv}(F_{SHIFT}, n_{dim})$$

where $M_{SHIFT}$ is the predicted CSVs and $n_{dim}$ denotes the dimension of raw radar detection points.

### C. Performance index

Experiments on RadarScenes are performed to evaluate the effectiveness of the proposed strategies. The spatial coordinates, velocities (compensated) and RCS values are used as inputs, while mean coverage (mCov) and mean average precision with the IoU threshold of 0.5 (mAP$_{0.5}$) on original detection points are reported for final instance prediction.

The mCov can be calculated by

$$mCov(G, P) = \frac{1}{|G|} \sum_{g \in G} \max_{p \in P} \text{IoU}(g, p)$$ (6)

where $G$ is the set of ground truth instances; $P$ is the set of predicted instances; $g$ and $p$ denotes the element (a set of points belonging to an instance) of sets $G$ and $P$, respectively. $\text{IoU}(g, p)$ is defined in \cite{frezzi2021radar3d}.

**Remark 4:** $\text{IoU}$ reflects the accuracy for a certain instance and its prediction, and $mCov$ is the average accuracy for all instances. If $mCov$ is close to 1, then it indicates that the performance of the model is superior.

For classification tasks, there are four relationships between the predicted class and the ground truth, and we can summarize them into Table I.

The precision and recall are defined by

$$\text{precision} = \frac{TP}{TP + FP}, \quad \text{recall} = \frac{TP}{TP + FN}.$$ (7)

Average precision (AP) is a metric for segmentation or detection that takes both precision and recall into consideration. It first sorts the predictions by their confidence scores (usually the predicted probability of this class) from large to small, then calculates the IoU between each prediction and all ground truths. If an IoU between a prediction and a ground truth is larger than a certain threshold, the prediction is marked as TP. After that, each prediction is traversed in order, and a precision-recall curve is drawn.

For example, a sorted and marked list of predictions for an instance segmentation task is $[TP, TP, FP, TP, FP, TP]$, and there are 5 objects belonging to this class in ground truth. If we look at the 3rd prediction, the precision can be calculated as $2/3$ because there are 2 TPs in the first 3 predictions; the recall can be calculated as $2/5$ because there are 2 TPs while
5 instances exist. The precision and recall of each prediction is listed in Table III.

Based on the precision and recall, a precision-recall (PR) coordinate system can be established whose horizontal axis is recall and vertical axis is precision. Then the PR curve connecting all points can be drawn, and the area enclosed by the PR curve and the axes is calculated. The PR curve shows the variation of precision and recall when the confidence score threshold changes (predictions with a lower confidence score will be discarded before calculating precision and recall). The area below the PR curve indicates the “average” performance of predictions of this class (or equivalently, AP). However, in practice, the longitudinal coordinate of a point is usually adjusted as the maximum value of that of points on its right. Fig. 7 illustrates the original and the adjusted PR curve of the previous example.

At last, we calculate the average of APs for all classes, and obtain the mAP. Larger mAP indicates the better performance.

D. Experimental Results and Analysis

Table III presents results of baseline and our two strategies without visual MLP enhancement. The performance of the SGPN based end-to-end instance segmentation improves 3% by modifying its loss function from $\ell_2$ loss to BCE loss. However, such deep learning based end-to-end instance segmentation only outperforms the baseline with limited improvements, but its inference time increases. In contrast, the semantic segmentation based clustering strategy reaches the mCov at 82.21% and mAP at 77.96% without significantly increasing inference time, and further improvement can be obtained by adding a CSVs prediction branch to PointNet++ using CS and NIP loss to achieve 82.78% mCov and 79.38% mAP.

**Remark 5**: It should be noted that both number of parameters and inference times for all models are given, and the inference time is defined as the average time cost when performing instance segmentation on test dataset using CPU.

The experimental results of the enhanced models are summarized in Table IV. It can be seen that all models in Table III and Table IV occupy from less than 1MB to 2MB storage space, making the strategies feasible for embedded radar based perception systems. To show the effectiveness of our proposed semantic segmentation based clustering with gMLP enhancement strategy, we use external attention and self attention for semantic segmentation network for comparison. Comparing with its original version, the gMLP-based model performs better in mCov and mAP by approximately 6%, while the self attention and external attention based model only improve about 3% and 2%, respectively, and with similar inference time and number of parameters. Notably, by attaching a tiny attention module to the spatial gating unit in gMLP, such model (called aMLP) further improves performance and slightly increases the inference time and the number of parameters. For SGPN based end-to-end instance segmentation, gMLP outperforms the baseline by approximately 5%, but its inference time increases dramatically, making the gMLP enhancement of such strategy infeasible in practice. Compare
gMLP enhanced end-to-end instance segmentation strategy with gMLP enhanced semantic segmentation based clustering strategy, it is witnessed that the latter outperforms the former significantly in either mCov, mAP, or inference time, indicating that only semantic segmentation based clustering with gMLP/aMLP enhancement strategy leads to the best performance in instance segmentation while still maintains the inference time and number of parameters fairly low.

Typical examples of instance segmentation results by using different strategies is visualized in Fig. 8. It is clear that the baseline strategy clustering based classification might obtain the incorrect instance and semantic estimation, while the deep learning based end-to-end instance segmentation strategy and its gMLP enhancement could correct the estimation partly, but they also generate other improper predictions. Comparatively, the semantic segmentation based clustering provides better estimation and its gMLP enhancement achieves 100% correct prediction for this particular case. Another result of consecutive frames by using the proposed semantic segmentation based clustering with gMLP enhancement could be seen in Fig. 9 where most of the instances could be segmented perfectly, even though some of them are spatially close with each other.

**Remark 6:** However, it can be also observed that two instances of cars are grouped together on the 2nd frame, where one car instance is divided into two instances, and another car instance is recognized as a large vehicle on 3rd frames. Such issues could be potentially solved by incorporating the consistent information from consecutive frames. For example, a car identified at the previous frame should not be recognized as a pedestrian in the current frame.

Although semantic segmentation based clustering with gMLP enhancement could provide acceptable storage size, it is still possible to compress it further, providing flexibility for allocating more powerful tracking algorithms in the entire radar perception system. Table V shows the comparison of results of gMLP enhanced semantic segmentation model with two different compressed approaches (one uses group convolution instead of conventional convolution in PointNet++) and the other reduces the dimension of the input of spatial gating units in gMLP blocks) and other enhanced semantic segmentation models. The compressed methods are capable of reducing the number of parameters and the corresponding storage memory consumption, and it can be seen that even the compressed gMLP-based model performs better than other enhanced models with comparable or even less memory consumption.

### V. Conclusion

Two strategies are proposed in this paper for radar point cloud instance segmentation. One is end-to-end instance segmentation using SGPN, and the other is semantic segmentation based clustering by PointNet++ and DBSCAN. The latter strategy provides better performance in mCov and mAP and faster inferring rate. Compared to the baseline clustering and classification, the inference time of the latter method does not significantly increase, whereas its performance is superior.

An enhancement with gMLP/aMLP is introduced by tuning the model parameters and the loss functions. The gMLP/aMLP enhanced semantic segmentation based clustering can provide improvements in mCov and mAP, whereas the inference time

| Methods | mCov(%) | mAP_0.5(%) | #Params/Memory | Inference Time |
|---------|---------|------------|----------------|----------------|
| Baseline: Clustering & Classification | DBSCAN + Random Forest Classifier | 79.54 | 76.09 | - / - | 17.1ms |
| End-to-End Instance Segmentation | SGPN (I2 loss for CF) | 77.32 | 73.21 | 75.8K/0.326MB | 63.8ms |
| Semantic Segmentation & Clustering | SGPN (our BCE loss for CF) | 79.91 | 76.15 | 75.2K/0.320MB | 27.4ms |
| | PointNet++ + DBSCAN | 82.21 | 77.96 | 75.2K/0.320MB | 27.4ms |
| | PointNet++ (with CSV head, I2 loss) + DBSCAN | 83.38 | 78.17 | 75.6K/0.324MB | 28.7ms |
| | PointNet++ (with CSV head, CS&NIP loss) + DBSCAN | 82.78 | 79.38 | 75.6K/0.324MB | 28.7ms |

| Methods | mCov(%) | mAP_0.5(%) | #Params/Memory | Inference Time |
|---------|---------|------------|----------------|----------------|
| End-to-End Instance Seg. | gMLP based SGPN | 79.91 | 76.15 | 75.8K/0.326MB | 63.8ms |
| Semantic Seg. based Clustering | PointNet++ with CSV head + DBSCAN | 82.78 | 79.38 | 75.6K/0.324MB | 28.7ms |
| | gMLP based PointNet++ with CSV head + DBSCAN | 88.54 | 85.24 | 339.9K/1.346MB | 336.3ms |
| | aMLP based PointNet++ with CSV head + DBSCAN | 89.53 | 86.97 | 435.0K/1.714MB | 35.4ms |
| | External Attention based PointNet++ with CSV head + DBSCAN | 85.23 | 81.41 | 122.7K/0.507MB | 31.2ms |
| | Self Attention based PointNet++ with CSV head + DBSCAN | 85.85 | 82.32 | 218.2K/0.876MB | 32.0ms |

| Methods | mCov(%) | mAP_0.5(%) | #Params/Memory | Inference Time |
|---------|---------|------------|----------------|----------------|
| Semantic Seg. based Clustering | gMLP based PointNet++ (Group Conv) + DBSCAN | 86.71 | 82.47 | 273.0K/1.090MB | 32.2ms |
| | gMLP (Reduced Dimension) based PointNet++ + DBSCAN | 86.36 | 82.68 | 179.0K/0.731MB | 32.0ms |
| | External Attention based PointNet++ with CSV head + DBSCAN | 85.23 | 81.41 | 122.7K/0.507MB | 31.2ms |
| | Self Attention based PointNet++ with CSV head + DBSCAN | 85.85 | 82.32 | 218.2K/0.876MB | 32.0ms |
only increases slightly compared to that without gMLP. The requirement of storage space for the enhanced model is less than 2MB, such that the gMLP enhanced semantic segmentation based clustering strategy is feasible for the real-time embedded radar-based ADAS/AD product. The storage space can be reduced even further, and the proposed method still maintains performance of instance segmentation by applying the light-weight approaches, indicating that more flexibility in designing radar-based perception can be achieved.

It should be admitted that, only detection points within single frame are used for training and predicting the instance information, even if the dataset provides sequences of radar point clouds. Taking point clouds of several consecutive frames simultaneously would possibly enable the model to extract features with consistent information, and facilitate the learning process. This will be included in our future research.

REFERENCES

[1] J. Dickmann, M. Klappstein, M. Hahn, N. Appenrodt, H. L. Bloecher, K. Werber, and A. Sailer, “Automotive radar: From detection and ranging to environmental understanding,” 2016 IEEE Radar Conference (RadarConf), pp. 1–6, 2016.

[2] C. Waldschmidt, J. Hasch, and W. Menzel, “Automotive radar — from first efforts to future systems,” IEEE Journal of Microwaves, vol. 1, pp. 135–148, 2021.

[3] S. Chen, B. Liu, C. Feng, C. Vallespi-Gonzalez, and C. K. Wellington, “3D point cloud processing and learning for autonomous driving,” ArXiv, vol. abs/2003.00601, 2020.

[4] C. Qi, H. Su, K. Mo, and L. Guibas, “3D point cloud classification and segmentation,” 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 77–85, 2017.

[5] C. Qi, L. Yi, H. Su, and L. Guibas, “Pointnet++: Deep hierarchical feature learning on point sets in a metric space,” in NIPS, 2017.

[6] W. Wang, R. Yu, Q. Huang, and U. Neumann, “Sgpn: Similarity group proposal network for 3D point cloud instance segmentation,” 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 2569–2578, 2018.

[7] S. Chen, J. Fang, Q. Zhang, W. Liu, and X. Wang, “Hierarchical aggregation for 3D instance segmentation,” ArXiv, vol. abs/2108.02350, 2021.

[8] M.-H. Guo, Z.-N. Liu, T.-J. Mu, and S. Hu, “Beyond self-attention: External attention using two linear layers for visual tasks,” ArXiv, vol. abs/2105.02358, 2021.

[9] I. Tolstikhin, N. Houlsby, A. Kolesnikov, L. Beyer, X. Zhai, T. Unterthiner, J. Yung, D. Keysers, J. Uszkoreit, M. Lucic, and A. Dosovitskiy, “Mlp-mixer: An all-mlp architecture for vision,” ArXiv, vol. abs/2105.01601, 2021.

[10] H. Liu, Z. Dai, D. R. So, and Q. V. Le, “Pay attention to mlps,” ArXiv, vol. abs/2105.08050, 2021.

[11] O. Schumann, M. Hahn, N. Scheiner, F. Weisheiu, J. F. Tilly, J. Dickmann, and C. Wöhler, “RadarScenes: A real-world radar point cloud data set for automotive applications,” ArXiv, vol. abs/2104.02493, 2021.

[12] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, “A density-based algorithm for discovering clusters in large spatial databases with noise,” in KDD, 1996.

[13] R. Zhang and S. Cao, “Robust and adaptive radar elliptical density-based spatial clustering and labeling for mmwave radar point cloud data,” 2019 53rd Asilomar Conference on Signals, Systems, and Computers, pp. 919–924, 2019.

[14] T. Wagner, R. Feger, and A. Stelzer, “Modification of dbscan and application to range/doppler/doa measurements for pedestrian recognition with an automotive radar system,” 2015 European Radar Conference (EuRAD), pp. 269–272, 2015.

[15] D. Birant and A. Kut, “Si-dbscan: An algorithm for clustering spatial-temporal data,” Data Knowl. Eng., vol. 60, pp. 208–221, 2007.

[16] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby, “An image is worth 16x16 words: Transformers for image recognition at scale,” ArXiv, vol. abs/2010.11929, 2021.

[17] X. Dong, J. Bao, D. Chen, W. Zhang, N. Yu, L. Yuan, D. Chen, and B. Guo, “Cswin transformer: A general vision transformer backbone with cross-shaped windows,” ArXiv, vol. abs/2107.00652, 2021.

[18] Z. Liu, Y. Lin, Y. Cao, H. Hu, Y. Wei, Z. Zhang, S. C.-F. Lin, and B. Guo, “Swin transformer: Hierarchical vision transformer using shifted windows,” ArXiv, vol. abs/2103.14030, 2021.
[19] A. Vaswani, N. M. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, “Attention is all you need,” ArXiv, vol. abs/1706.03762, 2017.

[20] S. Chen, E. Xie, C. Ge, D. Liang, and P. Luo, “Cyclemlp: A mlp-like architecture for dense prediction,” ArXiv, vol. abs/2107.10224, 2021.

[21] N. Scheiner, N. Appenrodt, J. Dickmann, and B. Sick, “A multi-stage clustering framework for automotive radar data,” 2019 IEEE Intelligent Transportation Systems Conference (ITSC), pp. 2060–2067, 2019.

[22] N. Scheiner, N. Appenrodt, J. Dickmann, and B. Sick, “Radar-based feature design and multiclass classification for road user recognition,” 2018 IEEE Intelligent Vehicles Symposium (IV), pp. 779–786, 2018.

[23] N. Scheiner, O. Schumann, F. Kraus, N. Appenrodt, J. Dickmann, and B. Sick, “Off-the-shelf sensor vs. experimental radar - how much resolution is necessary in automotive radar classification?” 2020 IEEE 23rd International Conference on Information Fusion (FUSION), pp. 1–8, 2020.

[24] A. Danzer, T. Griebel, M. Bach, and K. Dietmayer, “2d car detection in radar data with pointnets,” 2019 IEEE Intelligent Transportation Systems Conference (ITSC), pp. 61–66, 2019.

[25] A. Danzer, T. Griebel, M. Bach, and K. Dietmayer, “2d car detection in radar data with pointnets,” 2019 IEEE Intelligent Transportation Systems Conference (ITSC), pp. 61–66, 2019.