A Neural Network Based System for Efficient Semantic Segmentation of Radar Point Clouds

Alessandro Cennamo¹,² · Florian Kaestner¹ · Anton Kummert²

Accepted: 24 May 2021 / Published online: 29 May 2021
© The Author(s) 2021

Abstract
The last decade has witnessed important advancements in the field of computer vision and scene understanding, enabling applications such as autonomous vehicles. Radar is a commonly adopted sensor in automotive industry, but its suitability to machine learning techniques still remains an open question. In this work, we propose a neural network (NN) based solution to efficiently process radar data. We introduce RadarPCNN, an architecture specifically designed for performing semantic segmentation on radar point clouds. It uses PointNet++ as a building-block—enhancing the sampling stage with mean-shift—and an attention mechanism to fuse information. Additionally, we propose a machine learning radar pre-processing module that confers the network the ability to learn from radar features. We show that our solutions are effective, yielding superior performance than the state-of-the-art.

Keywords Radar · Point clouds · Neural network · Autonomous vehicle

1 Introduction

Lately, sensors such as camera, radar and lidar have been progressively more involved in the automotive industry [3]. The ability of Convolution Neural Networks (CNNs) to efficiently consume structured data like images have pushed the development of camera-based solutions, thus enabling a broad variety of applications [13,18,30]. However, key aspects to achieve fully-autonomous driving capabilities are robustness to weather conditions (i.e. rain, fog) and sensor redundancy. Lidar and radar can fill these requirements, but they produce data often represented as point clouds: an unstructured data-format that CNNs struggle to process [29]. Hence, over the last years, researchers have developed novel deep learning (DL) solutions to consume point clouds.
The most straightforward solution consists of transferring the point cloud data into 3D-grids. Indeed, by encoding point-features into voxels, the input becomes structured, hence fit to be consumed by conventional CNN-based methods [5,31]. Multi-view representation techniques take a similar approach: they use as input a collection of snapshots from the point cloud. Then, the resulting images are processed with different CNNs [9,26]. Grid-based approaches have dominated this area, achieving state-of-the-art performance on various tasks. However, the process of structuring the data results in either high computational costs or loss of information. Point-wise processing techniques overcome these limitations. The adoption of order-invariant networks enables the consumption of raw point cloud, thus without altering its information-content. However, as the input consists of a list points, its lack of structure hardens the extraction of spatial information. Nevertheless, several researchers have successfully designed architectures showing state-of-the-art performance [14,21,22]. Finally, their ability to maintain a low-cost profile has made point-wise techniques a competitive alternative to grid-based ones.

Although the last decade has seen large-scale adoption of radar in the automotive industry [1,6], complex tasks like object detection in crowded environments, last unsolved. DL techniques are known to be very data-demanding and the limited availability of public radar datasets has slowed down advancements in this area. Moreover, albeit very low-cost, automotive radars are unique sensors. This aspect limits the generalization of techniques devised for other domains. For instance, grid-based approaches designed for lidar data would result in either sparse cells or poor resolution, due to the higher sparsity of radar data [10]. In light of this, we think that point-wise processing methods are better-suited for the radar domain. In particular, we expect their ability to take advantage of all the input information to play a crucial role.

We believe that leveraging intrinsic properties of the radar sensor is of prime necessity for an efficient processing. In this work, we propose to enrich the input point-signatures by leveraging radar object-related features. Therefore, we design a pre-processing module that facilitates the final segmentation task by computing new per-point features. This module performs its task by means of a shared fully-connected (FC) network and can be easily plugged into any other existing point-wise approach. Herein, we show that the proposed pre-processing module helps PointNet++ [23] to improve its performance on our radar dataset. Additionally, we develop RadarPCNN, a novel architecture designed to perform semantic segmentation on raw radar point clouds. We leverage the pre-processing module introduced in this work. Moreover, we propose to improve usage of spatial information by means of the mean shift clustering algorithm [11]. To directly consume point clouds, PointNet++ [21] is used as a feature extractor module. In particular, two mini-PointNet++ are deployed to learn different features at object-level. Finally, our scheme uses an attention mechanism to confer the network the ability to filter the learned features. RadarPCNN proves superior performance to PointNet++, showing state-of-the-art semantic segmentation results on radar data.

This report extends our work in [4] with comprehensive studies about grouping process, usage of the radar Doppler feature and network error sources. Notably, we show that the network has the means to filter a non-ideal grouping process and the radar Doppler enables the network to achieve superior performance. The remainder of the document is structured as follows. Section 2 reviews the related works. Section 3 introduces our method. Section 4 describes the setup of our experiments ans Sect. 5 presents our results. Finally, Sect. 6 reports the conclusion drawn and address future work directions.
2 Related Works

Point clouds enable accurate perception of complex 3D scenes. For instance, by solving the semantic segmentation task it is possible to classify every input point according to a set of classes. Formally, given an unordered collection of \( N \in \mathbb{N} \) points \( x^i \in \mathbb{R}^m, i = 1, \ldots, N \) whose elements are the 2D/3D spatial coordinates plus features, one can define the segmentation task as the problem of estimating a function \( f: \mathbb{R}^{N \times m} \rightarrow \mathbb{R}^{N \times d} \) such that every input point is mapped to one of \( d \) semantic classes.

2.1 Deep Learning on Point Clouds

DL has proven particularly well-suited for processing low-level data, notably images. Yu et al. \[28\] used a CNN to perform place recognition. SPRNet \[27\] is a single-stage NN that can simultaneously perform object detection and instance segmentation. Esteva et al. \[8\] developed a NN-based system that can diagnose cancer from skin images. The authors of \[19\] trained a CNN to emulate human conversations. Yet, despite DL has been widely used for a variety of tasks, the unstructured nature of point clouds limits the application of standard techniques like CNNs or FC-networks.

To process raw point clouds, a network has to be invariant to order. Indeed, it is possible to produce a different input by shuffling the point order. However, the new input would contain the same point cloud—describing the same scene/object—, therefore, it is important that the algorithm would produce the same output. PointNet \[22\] is a milestone in this area. The authors noticed that it is possible to achieve an order-invariant network by using a combination of shared-FC layers and max-pooling: the former to extract point features and the latter to aggregate a global signature. However, despite effectively consuming point clouds, PointNet fails to use information of spatial proximity between points.

To solve the drawback of PointNet, the same authors developed PointNet++ \[21\]. This architecture is based on the observation that traditional CNNs owe their success to the capacity of hierarchically abstracting low-level structures into high-level patterns. Therefore, PointNet++ follows a hierarchical encoder-decoder structure: set-abstraction (SA) layers divide the input in overlapping cluster and extract a local descriptor, while feature-propagation (FP) layers interpolate high-level features into low-level point locations. PointNet is adopted to extract features from group of points. Among point-wise processing architectures, PointNet++ represents the current state-of-the-art.

Li et al, followed a different approach \[14\]. They found out that CNNs can process point clouds, provided some ordering. Therefore, they proposed to target an order-equivariant network, by weighting and permuting the input with a learned \( \mathcal{X} \) matrix. The authors devised \( \mathcal{X} \)-Conv: a module that performs \( \mathcal{X} \)-transformation on group of points followed by standard convolutions. Finally, they developed PointCNN, a PointNet++-like architecture, using \( \mathcal{X} \)-Conv as feature extractor module. The authors proved the effectiveness of the \( \mathcal{X} \)-transformation, showing state-of-the-art results on a broad variety of tasks.

2.2 Deep Learning on Radar Data

Radar is a low-cost sensor particularly widespread in the automotive industry, due to its ability to work in difficult weather condition and measure velocity.

Radar senses the environment by means of electromagnetic waves: the sensor radiates a signal in the surrounding environment, which is reflected back by the presence of potential
objects and collected by the sensor. The continuous-time received signal contains information about the number of objects in the scene, their distance from the sensor and other object-related attributes like relative radial velocity (Doppler frequency shift) and reflected power (proportionate to radar-cross section or RCS). To extract such features, the signal is discretized and processed with Fast Fourier Transformations, thus generating 3D-grids with information of intensity (RCS) for every range-Doppler-angle cell. Finally, it is possible to generate point clouds from such maps by means of the Constant False Alarm Rate algorithm, a thresholding method capable of detecting targets from clutter/noise. Therefore, radar point clouds have two features (Doppler and RCS) along with the usual spatial coordinates, thus enabling new applications like instantaneous detection of moving objects.

To reduce the sparsity of radar point clouds, a common approach involves the integration of radar reflections over time, so to generate occupancy or amplitude grid-maps [24]. These maps consist in structured, image-like inputs—whose cells contain information about occupancy or measured RCS—, thus enabling the use of standard CNNs [16,17]. However, despite this approach is well-suited for static-objects, it fails on dynamic ones, as the object velocity needs to be tracked to avoid generating a tail of reflections. Another classical radar processing approach consists in three steps: clustering, extraction of hand-crafted features and classification [20]. Wöhler et al. [25] designed a method which uses DBSCAN [7] at the clustering stage. Then, they leveraged the features of points in the same cluster to compute signatures which are used to infer the cluster’s class. Finally, the authors performed classification with a long-short term memory (LSTM) network [12], showing superior performance than the random forest algorithm [2]. Their results also showed that spatial and Doppler-related features are the most important for the classification task.

As radar is a widely-used sensor in the industry, there is a lack of open-source, publicly available datasets, thus complicating the development of new radar-specific DL solutions. For this reason, in the last years, researches have proposed to process radar point clouds with conventional methods (described in Sect. 2.1). Schumann et al. proposed to classify radar reflections as belonging to one of six classes: static target, truck, pedestrian, car, bike and pedestrian group [23]. They used a PointNet++ network [21] to consume raw point clouds. The authors showed that such technique can successfully process radar point clouds, though it has been designed for processing much denser inputs. Feng et al. addressed the classification problem of vehicle versus guardrail [10]. They also used PointNet++ to perform the task. Moreover, they showed that the network performance can be enhanced by the introduction of hand-crafted statistics in the clusters’ signature (such as point density).

3 Our Method

Radar point clouds differ from lidar ones as they hold object-related information in point-features. Classical radar-processing methods usually leverage this knowledge indirectly: firstly, they group together points belonging to the same object, then, they extract hand-crafted statistics from point-features [20]. However, to effectively consume such data, we find that one cannot refrain from fully exploiting the object-specific information carried by each data-point.

DL networks possess the ability to cast low-level features into meaningful high-level representations. Therefore, we believe that NNs can help to extract informative per-point descriptors from radar measurements. Herein, we propose to generate new point-signatures via a pre-processing module that leverages single-point radar features. Specifically, given the
generic \(i\)-th point, our module combines the input features \(x^i\)—containing \(x\), \(y\), Doppler and RCS—by means of a shared-FC network with trainable parameters \(P\), as in Eq. 1.

\[
x_p^i = f_p(x^i; P) = \text{Relu}(P_3^T \text{Relu}(P_2^T \text{Relu}(P_1^T x^i + p_1) + p_2) + p_3)
\]

Figure 1 reports the operations. Notice that all points share the same parameter-set \(P\). Yet, every point produces a unique signature, driven by the corresponding input features. Our module is designed to preserve the input spatial structure: its output consists in a point cloud with unaltered locations and enriched features. As a result, this module enables the network to leverage the radar view of the objects, encoding the input features into a latent high-dimensional representation that increases the distance between points with diverse radar characteristics. In addition, considering its simplicity, we believe that any point-wise processing architecture would benefit from the introduction of our module.

PointNet++ [21] represents the state-of-the-art solution for consuming raw point clouds, as it can encode spatial structural information into high-level object descriptors. However, we find this architecture to be a sub-optimal approach for processing radar data. Indeed, radar usually provides point clouds with low density, containing poor structural information, thus limiting the performance of PointNet++. Therefore, we develop RadarPCNN, a network specifically-designed for radar point clouds that operates at object level. In our implementation we favor solutions that enhance the usage of spatial structural information, as we believe it to be a key aspect for efficient processing.

RadarPCNN is reported in Fig. 2. It adopts our pre-processing module to enrich the input feature-set. Then, the deep latent point cloud is fed to an altered PointNet++ architecture, so that our network can inherit the ability to learn from structural information. As described in Sect. 2.1, PointNet++ uses SA encoder layers and FP decoder layers. SA layers perform three operations: sampling, grouping and feature extraction. Specifically, given a point cloud \(\mathcal{X} = \{x^i \in \mathbb{R}^m, i = 1, \ldots, N\}\), the sampling process selects a collection of points \(\mathcal{X}' \subset \mathcal{X}\), the grouping process clusters together points \(x^j \in \mathcal{X}'\) that lie in the neighborhood\(^1\) of each point \(x^j \in \mathcal{X}'\) and the feature extraction process encodes the clusters’ structural information into a single point-descriptor. The outcome of a SA layer is a point cloud with locations defined by the points \(x^j \in \mathcal{X}'\) and features computed as

\[
x_S^j A = \text{MaxPool}(\text{Relu}(W_S A^T G^j + B_S A))
\]

where \(G^j\) is the matrix whose columns contain the points \(x^i \in \mathcal{X}\) that lie in the neighborhood of \(x^j\) and \(W_S A\) and \(B_S A\) learnable parameters. In our architecture, we decide to tamper the sampling process by using mean shift (MS) in place of the original farthest point sampling (FPS) method. As MS focuses on high density points rather than sampling uniformly, we

\(^1\) The neighborhood area is defined by a distance function and a search range.
assume that it will improve the usage of spatial information. Specifically, we sum up Gaussian kernels (with given bandwidth) centered at each point location and use local maxima of the resulting distribution to sample points for $X'$. Notice that we involve MS in the sampling process, without altering the grouping stage. Furthermore, we decide to use two PointNet++ with single SA-FP layer in place of the original hierarchical structure. Besides limiting the hidden-layers point-sparsity, this configuration allows us to design two object-oriented sub-networks, seeking for small and large objects, as shown in Fig. 2. The top branch samples densely and groups points in small areas (2 meters), while the bottom one uses a sparser sampling strategy and larger neighborhood regions (4 to 8 meters). The MS bandwidth is the parameter that controls the density of the sampling process. In our implementation, we adopt unaltered FP layers, performing distance weighted interpolation followed by shared-FC layers.

For every input point, each of the two PointNet++ produces a different descriptive pattern. To fuse these information into a single point-signature, we design the attention mechanism in Fig. 3. It involves a multi-layer shared FC-network, to compute the attention weights, and a weighted combination, to merge the two feature vectors. In more details, given the $i$-th point and the features $x_1^i$ and $x_2^i$ learned, respectively, from top and bottom branch (green and red points), we compute the attention weights as

$$w_1^i = f_a(x_1^i; A, a) = \sigma\left(a^T \text{Relu}(A^T x_1^i)\right)$$ and $$w_2^i = f_a(x_2^i; A, a)$$

(3)
where \( A \) and \( a \) are trainable parameters and \( \sigma(\cdot) \) the sigmoid function. Notice how both feature vectors share the same network parameters. Finally, \( x_i^1 \) and \( x_i^2 \) are merged into a single classification vector \( x_o^i \) (orange point) by means of a weighted combination, as illustrated in Eq. 4.

\[
x_o^i = w_1^i \cdot x_1^i + w_2^i \cdot x_2^i
\] (4)

Our attention mechanism enables the network to perform a smart combination. By computing the weights as a function of the feature vectors, indeed, it is possible to increase the focus on the vector with more relevant information. Yet, the network can leverage knowledge extracted from both PointNet++ branches.

The point-signatures produced by the attention mechanism are finally fed to a shared FC-network to generate the output prediction scores. We train every module in an end-to-end fashion without any intermediate loss, as our architecture results in a fully-differentiable function.

4 Experimental Setup

In this work, we target the segmentation task between two semantic classes: moving vehicle and moving pedestrian. We measure performance in terms of precision, recall and \( F_1 \) score on the positive classes along with their macro-average version (with each class contributing equally).

4.1 Dataset

To evaluate our solutions, we collected data using vehicles equipped with six frequency-modulated continuous-wave (FMCW) 77GHz radars (sides, front and back corners). The vehicles recorded urban scenes under clear, rainy and cloudy weather conditions. The sensors were able to collect reflections from targets up to \( \pm 90 \) meter, with a field-of-view of \( \pm 75^\circ \). Totally, our dataset provides 84 video sequences and more than 40 million points, annotated as pedestrian, vehicle or other/clutter. Without involving occupancy-grid maps, automotive radars provide poor information about stationary objects. Beside sharing the same Doppler with other still instances (e.g. buildings), they do not produce enough per-object points to enable the extraction of shape information. Therefore, we decide to focus on moving instances. In this setup, it is critical to define when an object is considered moving. We opt for different thresholds: a pedestrian is deemed moving if its ground-truth velocity is above 0.5 m/s, whilst for vehicles we use 2.5 m/s. Notice that these thresholds are very sensitive parameters, as they can either create a trivial problem or mislabel stationary objects as moving. However, during our experimentation, we found out that the network would correctly classify points from objects with low velocity as moving, regardless of the threshold-settings. Thus, we mask reflections with velocity below the selected thresholds in order to not impact the training nor the evaluation process. Nevertheless, we treat point from objects that does not move at all in the scene (e.g. parked cars) as other/clutter. Finally, we split the dataset in train and test set as shown in Table 1, to get similar distributions of classes and weather conditions.
Table 1  Data distribution: percentage and total number of points

| Set     | Moving pedestrian | Moving vehicle | Other and clutter |
|---------|-------------------|----------------|-------------------|
| Train   | 2.26% (721 865)   | 6.12% (1 955 937) | 91.62% (29 282 877) |
| Test    | 1.40% (142 647)   | 5.63% (574 321)  | 92.98% (9 493 667) |

4.2 Settings

The video sequences contained in our dataset are divided in frames. As each frame provides only few hundreds of points, similarly to [23], we decide to increase the context information by aggregating reflections from adjacent frames. We adopt a window of 200 ms and compensate the ego-movement to obtain the point location w.r.t the ego-position in the last frame. Furthermore, we refrain from using elevation measurements, due to the sensor low resolution. Instead, we use the ego-compensated Doppler value as z-coordinate—so as to increase the spatial distance between instances with different velocity—and the RCS as point feature. Finally, as we are interested in moving objects, we fix the input size to 1200 points and sample with importance using the ego-compensated Doppler feature. Thus, in case the input has more than 1200 points, reflections with large absolute Doppler values are more likely to be retained. Conversely, with less than 1200 points, we prefer to duplicate reflections with the same criteria. As a side effect, this process increases the positive classes population, thus reducing the class imbalance of the task, critical when training the model.

Concerning RadarPCNN, the pre-processing module uses three FC layers with 8, 16 and 32 channels, respectively. The number of representative points for the top/bottom branch of Fig. 2 are set to 500/150. We use FPS to select among the points sampled by MS or fill the gaps. Both PointNet++ use multi-scale grouping, using the Euclidean distance function with search areas of 1, 1.5 and 2 meters (top branch) and 4, 6 and 8 meters (bottom). The attention mechanism aggregates 128-dimensional point-signatures using a FC layers of 8, 4, and 4 channels. The final classification FC-network has three layers of size 256, 64 and 32, before the output layer. Dropout with rate 0.5 interleaves the classification network layers.

To deal with the class imbalance reported in Table 1, we train the network using focal loss [15]. In addition, we avoid further bias in the model by addressing two binary tasks (pedestrian vs non-pedestrian, vehicle vs non-vehicle) in place of a multi-class problem. The negative class is then trained from both the binary tasks. Sigmoid is used as output activation function. The class that achieves the highest score represents the final network prediction. The negative class is predicted when the final score does not exceed 0.5.

We set the hyper-parameters by performing five-fold cross validation on the train-set. As a result, the vehicle/pedestrian focal loss weight ($\alpha$ in [15]) is set to 0.85/0.9. Unless otherwise stated, all the networks tested use the same parameters for fair comparison. Finally, we train on the whole train-set for 20 epochs and select the best performing configuration. Then, we test it 10 times to average out the sources of randomness and report the mean values.

5 Results

Radar possesses the ability to measure radial velocity. Since we address the semantic segmentation task of moving instances, there is the legitimate doubt that a simple thresholding algorithm could suffice to solve it. To discredit this theory and prove that the task is not
trivial we decide to evaluate a random forest model [2] on our dataset. This ensemble-based method learns to classify the input by comparing its features with a set of thresholds. It is clear from Table 2 that random forest achieves significantly inferior performance w.r.t the other methods, thus suggesting that the problem is not obvious.

The state-of-the-art static architecture for processing raw radar point clouds is PointNet++, therefore, we decide to benchmark our solutions against it. We adopt the implementation in [23] with minor modifications to comply with our experimental setup. Specifically, we halve both the number of SA layer representative points and the width of the output FC network. This allows the model to encompass the reduced input dimensionality and number of output classes. Moreover, as the network has more parameters/FLOPS than ours, we experiment with a shallower version of PointNet++, reducing the number of SA/FP layers from three to two. In this way we enable fair comparison. Finally, we plug our pre-processing module to the shallow PointNet++ to assess its ability to generalize to other models.

Table 2 shows the results of our experiment. RadarPCNN achieves the best performance on our dataset, outperforming all the other network configurations tested. Remarkably, despite its higher capacity, PointNet++ cannot outperform our model. This means that our solutions enable RadarPCNN to perform a more efficient processing. In its shallow configuration, PointNet++ undergoes a predictable drop in performance. However, impressive results are achieved when plugging our pre-processing module: the network exhibits a boost in performance, challenging RadarPCNN. This behavior indicates that other point-wise architectures benefit from the enriched feature-set produced by our pre-processing module, enhancing their ability to learn from radar features. Yet, while the shallow PointNet++ with our module shows circa 15% more parameters, it cannot outperform RadarPCNN. This means that our architecture trades complexity with performance in a better way, thus processing the point cloud more efficiently.

From the confusion matrices in Fig. 4 it is possible to notice the distribution of errors. Notice how very few samples of the positive classes are confused with the wrong positive class. Additionally, notice how wrong prediction from the negative class affect the precision of the models. This is mostly due to the strong class imbalance present in our dataset, as the negative class is way more populated than the positive ones. Despite the countermeasures implemented during training, still the networks remain partially influenced by this effect.

5.1 Ablation Studies

5.1.1 Fusion Method

In this section we investigate the benefits of the proposed attention mechanism compared with different fusion methods. We implement addition, which sums-up the features learned for the same point, and concatenation, chaining the features to result in a double-sized signature vector. The results in Table 3 show that our attention mechanism performs the most efficient processing, demanding a fairly small amount of parameters to reach top performance.

5.1.2 Sampling Method

In our architecture, the sampling process is altered with the mean shift algorithm. This section provides an investigation on the effects of MS compared with the classical FPS method. Additionally, to gather more insights, we monitor the network behavior while varying the number of points sampled in the SA layers. FPS expects as input the amount of point to
Table 2  Semantic segmentation results on our dataset (%)

| Method                | Moving pedestrian | Moving vehicle | Average |
|-----------------------|-------------------|----------------|---------|
|                       | Prec.  | Recall | $F_1$ | Prec. | Recall | $F_1$ | Prec. | Recall | $F_1$ | Param. | FLOPs | Time |
| Random Forest         | 16.92  | 70.43  | 27.29 | 54.49 | 84.09  | 66.13 | 35.71 | 77.26  | 46.71 | –      | –     | 31.4$^a$ |
| PointNet++ [23]       | 46.10  | 75.77  | 57.33 | 72.77 | 82.89  | 77.50 | 59.44 | 79.33  | 67.41 | 418.6K | 1.55G | 31.2 |
| PointNet++ (shallow)  | 43.82  | 75.31  | 55.40 | 71.18 | 84.01  | 77.06 | 57.50 | 79.66  | 66.23 | 191.2K | 0.95G | 28.5 |
| PointNet++ (sh. + pr.)| 47.70  | 76.77  | 58.84 | 73.16 | 93.24  | 81.99 | 60.43 | 85.01  | 70.42 | 198.8K | 1.02G | 30.6 |
| RadarPCNN (ours)      | **48.64** | 75.82  | **59.27** | **75.78** | 91.44  | **82.88** | **62.21** | 83.63  | **71.07** | **175.3K** | 1.02G | **26.8** |

Inference time is computed on Nvidia GeForce RTX 2080 (ms)

$^a$CPU time
sample. Concerning MS, we reduce the number of cluster centers outputted by progressively widening the kernel bandwidth. Figure 5 plots the results.

Notice how MS (solid line) constantly outperforms FPS (dashed). However, the best setting (500/150) exhibits relatively small performance variations. We attribute this effect to the increasing similarity of the two algorithms while ramping-up the number of points to sample. To this end, notice that the output of MS with a kernel of infinitesimal bandwidth is exactly the same of FPS when asked to sample all the input points. On the other hand, it is impressive the different behavior of the curves while reducing the number of representative points. Here, MS considerably outperforms FPS, therefore, we can conclude that, by attending highly populated locations of the space, MS enhances the usage of point-proximity information. Vice versa, FPS is well-suited for dense sampling strategy, but progressively looses meaningful scene information as the number of representative points is reduced. Our consideration are further confirmed by the distinct attitude of the positive classes. Indeed, MS impact much more pedestrians than vehicles. We believe this is due to the fact that, because of their smaller dimensions, pedestrians generate fewer reflections. Therefore, besides MS will still consider small clusters, a reduction of the representative points, would critically reduce the probability of FPS sampling them. Finally, bearing in mind that more representative points traduces in higher computational costs, the favorable performance-complexity trade-off performed by MS makes it a promising candidate for embedded systems, where computation is a very crucial resource.

5.1.3 Grouping Method

RadarPCNN uses SA layers to encode information. As described in Sect. 3, it relies on three operations: sampling, grouping and feature extraction. In the previous paragraph, we studied the sampling process, while here we investigate different grouping techniques. In particular, we compare the conventional spherical method, with two ellipsoidal-based methods.

Spherical grouping uses a sphere to gather neighbor points around the sampled one. Formally, given the coordinates \((x^i, y^i, z^i)\) of the \(i\)-th point \(x^i \in X\) and the coordinates \((x^j, y^j, z^j)\) of the \(j\)-th sampled point \(x^j \in X'\), it selects those points belonging to \(X\) whose Euclidean distance

\[
d_{sph} = \sqrt{x^2 + y^2 + z^2}
\]

is lesser than the radius \(r\), with \(x = x^i - x^j\) and \(y, z\) computed accordingly.
Table 3  RadarPCNN performance using different fusion methods (%)

| Method  | Moving Pedestrian | Moving Vehicle | Average |
|---------|-------------------|----------------|---------|
|         | Prec.  | Recall  | $F_1$ | Prec.  | Recall  | $F_1$ | Prec.  | Recall  | $F_1$ | Param. | Time  |
| Conc    | 48.27  | 76.50   | **59.19** | 74.98  | 91.78   | 82.54 | 61.63  | **84.14** | 70.87 | 206.9K | 25.7  |
| Add     | 47.11  | 75.87   | 58.88  | 74.07  | **92.13** | 82.12 | 61.09  | 84.00   | 70.50 | **174.1K** | 25.5  |
| Att (ours) | **48.64** | 75.82   | **59.27** | **75.78** | 91.44   | **82.88** | **62.21** | 83.63   | **71.07** | 175.3K | 26.8  |

Inference time is computed on Nvidia GeForce RTX 2080 (ms)
The ellipsoidal grouping represents an alteration of the spherical method. We consider the possibility to learn to filter at the grouping stage, thus we define a generic ellipsoid by introducing additional parameters to equation 5. Specifically, we compute the point distance as

\[ d_{ell} = \sqrt{\alpha(x \cdot \cos \phi + y \cdot \sin \phi)^2 + \beta(x \cdot \sin \phi - y \cdot \cos \phi)^2 + \gamma z^2} \]  

with \( \alpha, \beta, \gamma \) and \( \phi \) parameters of the ellipsoid. In this way we allow a scaling of the Euclidean distance and a rotation in the \( XY \) plane. We decide not to rotate in the \( Z \) dimension as it is set to the Doppler velocity—the is no reason for a linear relation between point location and velocity. Using equation 6, we set-up two experiment: dynamic and static ellipsoid grouping. In the former we let the network regress the ellipsoid parameters, while in the latter we use a set of static configurations. In more details, the dynamic ellipsoid method first regress the 4 parameters using the spherical grouping method, then uses the estimated ellipsoid to group the points which are later used for encoding. The static method, instead, collects the neighborhoods using 5 configurations: the spherical distance (i.e. \( \alpha = \beta = 1, \phi = 0^\circ \)) plus four ellipsoids with parameters \( \alpha = 0.5, \beta = 1 \) rotated by 45° each. The \( Z \) axis does not experience any scaling (i.e. \( \gamma = 1 \)). Finally, we use an attention mechanism similar to the one in Fig. 3 to combine the features learned by every configuration. Both these methods enable the network to learn which points are the most relevant.

Table 4 contains the results. The standard spherical grouping method shows superior performance than the ellipsoid-based approaches. We believe the reasons to be twofold. On one hand, we use PointNet as feature extraction module which employs max-pooling to filter the features learned from each point. Thus, the network has already a means for selecting the most relevant neighboring points. On the other hand, the grouping operation is not a transformation of the input but rather a selection. Since this process does not involve any learned weights or functions, there is no direct gradient signal and the grouping process cannot be directly optimized. Hence, we conclude that the architecture does not need a perfect grouping step because it is able to filter the input by other means. Therefore, we opt to use the spherical grouping method in RadarPCNN, as it requires fewer parameters.
| Method          | Moving pedestrian | Moving vehicle | Average         | Param. | Time |
|-----------------|--------------------|----------------|-----------------|--------|------|
|                 | Prec.   | Recall  | $F_1$   | Prec.   | Recall  | $F_1$   | Prec.   | Recall  | $F_1$   |        |      |
| Dynamic ellipsoid | 44.70   | 71.94   | 55.14   | 66.97   | 89.68   | 76.64   | 55.83   | 80.81   | 65.89   | 191.6K | 28.1 |
| Static ellipsoid | 48.92   | 70.00   | 57.25   | 71.81   | 90.09   | 79.92   | 60.36   | 80.04   | 68.58   | 204.7K | 27.7 |
| Spherical       | 48.64   | **75.82** | **59.27** | **75.78** | **91.44** | **82.88** | **62.21** | **83.63** | **71.07** | **175.3K** | **26.8** |

Inference time is computed on Nvidia GeForce RTX 2080 (ms)
Table 5  RadarPCNN performance under different usage of the Doppler feature (%)

| Method         | Usage   | Moving pedestrian | Moving vehicle | Average |
|----------------|---------|--------------------|----------------|---------|
|                | Prec.   | Recall             | Prec.          | Recall  | Prec.   | Recall | F_1 | Prec.   | Recall | F_1 | Prec.   | Recall | F_1 |
| Doppler Feat   | 37.15   | 63.74              | 46.94          | 65.81   | 79.08   | 71.84  | 51.84 | 71.41   | 59.39   | 59.39 |
| Doppler Coord  | 42.97   | 70.67              | 53.37          | 69.65   | 79.89   | 74.42  | 56.31 | 75.28   | 63.89   | 63.89 |
| Comp. Doppler  | 46.96   | 71.03              | 56.54          | 74.16   | 91.13   | 81.62  | 60.55 | 81.08   | 69.08   | 69.08 |
| Comp. Doppler  | 48.64   | 75.82              | 59.27          | 75.78   | 91.44   | 82.88  | 62.21 | 83.63   | 71.07   | 71.07 |

5.1.4 Doppler Experiments

In this section we investigate how the network uses radar velocity measurements. Since every radar reflection has a measured Doppler—containing information of relative radial velocity—, it can be used as additional point feature. However, relative velocity information might be deceiving. Another approach, therefore, consists in recovering the absolute radial velocity, by compensation of the ego-vehicle motion. In this work, we decide to measure the ego-velocity with an odometry sensor and then compensate it by subtracting its radial component w.r.t. the point position. The resulting ego-compensated Doppler can then be used as point feature. Furthermore, since the sensors used in this work provide poor elevation measurements and velocity is a strong object descriptor, another option consists in using the Doppler as third spatial coordinate. Therefore, we report in Table 5 the performance of RadarPCNN while using relative or ego-compensated Doppler as point-feature or z-coordinate. As expected, the network benefits from the compensated Doppler, yielding an average improvement of circa 10% over the relative counterpart. Consider, indeed, an object moving at the same pace of the ego-vehicle that produces a null Doppler measurement. Since the network has no clue about the ego-velocity, in this case it cannot recover the true object (radial) velocity, thus becoming prone to predict the wrong class. As of its usage, Table 5 shows that incorporating the Doppler in the coordinates helps the network to achieve superior performance. We attribute this effect to the introduction of an additional spatial separation between points of objects with different velocities. To explain it, consider, for instance, a vehicle driving-by a parked car. The reflections from these two instances will have closeness in spatial XY dimension, but they will be far away in the velocity plane. Analyzing the inner operation of RadarPCNN, there are two processes that benefit from this effect: sampling and grouping. In the former, the additional separation reduces the probability of merging points from the moving and parked vehicles in the same clusters, hence enabling the network to sample representative points for both objects. In the latter, it reduces the probability that points from these two objects are found in the same group. Since the ego-compensated Doppler used as z-coordinate yields the best performance, we select this configuration in RadarPCNN.

5.2 Visual Analysis

We believe that visual analyses are very important as they allow gathering of fine details about the network behavior. In this section, therefore, we report insights collected from visualization of RadarPCNN predictions. Figure 6 contains a sample test frame (see caption). Notice how the network is able to correctly perform the semantic segmentation task, distinguishing between the provided classes. Another interesting finding in Fig. 6 comes from the yellow
Fig. 6 RadarPCNN predictions. Yellow/red points are moving vehicle/pedestrian predictions. Ground truth BBs follow the same color-map. Gray BBs belong to stationary vehicles, while orange BBs marks vehicles with absolute speed between 1 and 2.5 m/s. Left. Front/rear camera view. Right. 3D visualization. Best viewed in color.

points inside the orange bounding-box (BB): despite this vehicle travels at a speed inferior to the designed threshold, RadarPCNN classifies its reflections as belonging to a moving instance. Therefore, the network learns to predict moving objects independently from the threshold set during training.

5.2.1 Case of Study: Bushes as Pedestrians

During our visual study of the network, we found out that sometimes reflections from bushes are mistakenly classified as pedestrian. Figure 7 shows an example, where it is possible to notice the red points on the right of the ego-vehicle. From the camera images, it is clear that the points come from bushes. To understand whether this effect is peculiar to RadarPCNN, we repeat the same study with PointNet++ and we find out similar sources of error. This suggests that the error is not network-specific, therefore, we attribute it to the strong radar similarity between bushes and pedestrians. Indeed, bushes have different moving parts which might produce a fluctuation in the Doppler signature, as does pedestrians [20]. In order to confirm our assumption, we perform an additional experiment. We collect the Doppler value of bushes reflections that the network has mistakenly predicted as pedestrian. Then, we repeat it for reflection from stationary objects—we select buildings and parked vehicles. Figure 8 contains the results. It shows the histogram of Doppler values for the two different set of objects. Moreover, as a reference, we report the Doppler distribution of points labeled as pedestrians (black dotted line). As expected, stationary objects exhibit a very narrow Doppler spectrum, with values very close to 0 m/s. On the other hand, the plot reports a strong similarity between bushes and pedestrians: both show a wider spectrum, with an important component around 0 m/s. Additionally, in terms of spatial location, bushes are often found nearby static objects like buildings (see Fig. 7), where it is also likely to find pedestrians. This experiments proves that from a radar perspective bushes resemble pedestrians, hence explaining the reason of the network confusion. Finally, we believe that RCS represents a means for distinguishing these two instances, as bushes/trees are scarce reflectors, lacking a main body that pedestrians do have.
6 Conclusions

In this work, we demonstrate that it is possible to efficiently process radar point clouds with NN by leveraging the information contained in the point-features. Our pre-processing module dramatically improves the performance of PointNet++, thus proving well-suited for other point-wise processing techniques. Additionally, we introduce RadarPCNN, a NN capable to achieve superior performance by tackling intrinsic problem of radar data during the design phase. Indeed, we show that MS optimizes the usage of spatial information. Moreover, we find out that the SA layer does not require an ideal grouping process and that the ego-compensated Doppler drastically improves classification when used as spatial coordinate. Finally, we investigate RadarPCNN from a visualization perspective and we found out that bushes are easily confused as pedestrians, due to their radar similarity.

In future, the network performance can be further improved by using a tracker or involving temporal information, to sort out minor prediction error. Another interesting experiment consists in testing the effect of our pre-processing module on other point-wise processing methods. It would also be intriguing to study the performance of our solutions on differ-
ent radar datasets. Finally, a performance/cost comparison with grid-based methods would ultimately assess the effectiveness of the proposed solutions.

**Funding** Open Access funding enabled and organized by Projekt DEAL.

**Declarations**

**Conflict of interest** The authors declare that they have no conflict of interest.

**Open Access** This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article’s Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article’s Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit http://creativecommons.org/licenses/by/4.0/.

**References**

1. Alessandretti G et al (2007) Vehicle and guard rail detection using radar and vision data fusion. IEEE Trans Intell Transp Syst 8(1):95–105
2. Breiman L (2001) Random forests. Mach Learn 45(1):5–32. https://doi.org/10.1023/A:1010933404324
3. Caesar H et al (2019) nuscenes: a multimodal dataset for autonomous driving. CoRR
4. Cennamo A, Kästner F, Kummert A (2020) Leveraging radar features to improve point clouds segmentation with neural networks. In: Proceedings of the 21st EANN 2020 conference, Springer, pp 119–131. https://doi.org/10.1007/978-3-030-48791-1_8
5. Engelcke M, et al (2017) Vote3deep: fast object detection in 3d point clouds using efficient convolutional neural networks. In: 2017 IEEE international conference on robotics and automation, ICRA, pp 1355–1361. https://doi.org/10.1109/ICRA.2017.7989161
6. Eriksson LH, As B (1997) Automotive radar for adaptive cruise control and collision warning/avoidance. In: Radar 97 (Conf Publ No 449), pp 16–20. https://doi.org/10.1049/cp:19971623
7. Ester M et al (1996) A density-based algorithm for discovering clusters in large spatial databases with noise. In: Proceedings of the second international conference on knowledge discovery and data mining (KDD-96), pp 226–231
8. Esteva A, Kuprel B, Novoa RA, Ko J, Swetter SM, Blau HM, Thrun S (2017) Dermatologist-level classification of skin cancer with deep neural networks. Nature 542(7639):115–118. https://doi.org/10.1038/nature21056
9. Feng Y et al (2018) GVCNN: group-view convolutional neural networks for 3d shape recognition. In: 2018 IEEE conference on computer vision and pattern recognition, CVPR, pp 264–272. https://doi.org/10.1109/CVPR.2018.00035
10. Feng Z et al (2019) Point cloud segmentation with a high-resolution automotive radar. In: AmE 2019—automotive meets electronics; 10th GMM-symposium, pp 1–5
11. Fukunaga K, Hostetler LD (1975) The estimation of the gradient of a density function, with applications in pattern recognition. IEEE Trans Inf Theory 21(1):32–40. https://doi.org/10.1109/TIT.1975.1055330
12. Hochreiter S, Schmidhuber J (1997) Long short-term memory. Neural Comput 9(8):1735–1780. https://doi.org/10.1162/neco.1997.9.8.1735
13. Jose A et al (2019) A novel traffic sign recognition system combining viola–jones framework and deep learning. In: Soft computing and signal processing, Springer, pp 507–517
14. Li Y et al (2018) Pointcnn: convolution on x-transformed points. In: Advances in neural information processing systems 31: annual conference on neural information processing systems, NeurIPS, pp 828–838
15. Lin T et al (2017) Focal loss for dense object detection. In: IEEE international conference on computer vision, ICCV, pp 2999–3007. https://doi.org/10.1109/ICCV.2017.324
16. Lombacher J, Hahn M, Dickmann J, Wöhler C (2017) Object classification in radar using ensemble methods. In: 2017 IEEE MTT-S international conference on microwaves for intelligent mobility (ICMIM), pp 87–90
17. Lombacher J, Laudt K, Hahn M, Dickmann J, Wöhler C (2017) Semantic radar grids. In: 2017 IEEE intelligent vehicles symposium (IV), pp 1170–1175
18. Pintelas E et al (2020) Explainable machine learning framework for image classification problems: case study on glioma cancer prediction. J Imaging. https://doi.org/10.3390/jimaging6060037
19. Prakash A, Brockett C, Agrawal P (2016) Emulating human conversations using convolutional neural network-based ir. arXiv:1606.07056
20. Prophet R et al (2018) Pedestrian classification for 79 ghz automotive radar systems. In: 2018 IEEE intelligent vehicles symposium, IV, pp 1265–1270. https://doi.org/10.1109/IVS.2018.8500554
21. Qi CR et al (2017) Pointnet++: deep hierarchical feature learning on point sets in a metric space. In: Advances in neural information processing systems 30: annual conference on neural information processing systems, pp 5099–5108
22. Qi CR et al (2017) Pointnet: seep learning on point sets for 3d classification and segmentation. In: 2017 IEEE conference on computer vision and pattern recognition, CVPR, pp 77–85. https://doi.org/10.1109/CVPR.2017.16
23. Schumann O et al (2018) Semantic segmentation on radar point clouds. In: 21st International conference on information fusion, FUSION, pp 2179–2186. https://doi.org/10.23919/FUSION.2018.8455344
24. Werber K, Rapp M, Klappstein J, Hahn M, Dickmann J, Dietmayer K, Waldschmidt C (2015) Autonomous radar gridmap representations. In: 2015 IEEE MTT-S international conference on microwaves for intelligent mobility (ICMIM), pp 1–4
25. Wöhler C et al (2017) Comparison of random forest and long short-term memory network performances in classification tasks using radar. In: Sensor data fusion: trends, solutions, applications, SDF, pp 1–6. https://doi.org/10.1109/SDF.2017.8126350
26. Yang B et al (2018) Attentional aggregation of deep feature sets for multi-view 3d reconstruction. CoRR
27. Yu J, Yao J, Zhang J, Yu Z, Tao D (2021) Sprnet: single-pixel reconstruction for one-stage instance segmentation. IEEE Trans Cybern 51(4):1731–1742. https://doi.org/10.1109/TCYB.2020.2969046
28. Yu J, Zhu C, Zhang J, Huang Q, Tao D (2020) Spatial pyramid-enhanced netvlad with weighted triplet loss for place recognition. IEEE Trans Neural Netw Learn Syst 31(2):661–674. https://doi.org/10.1109/TNNLS.2019.2908982
29. Zaheer M et al (2017) Deep sets. In: Advances in neural information processing systems 30: annual conference on neural information processing systems, pp 3391–3401
30. Zhou X, Wingert T, Sauer M, Kundu S (2020) Development of a camera-based driver state monitoring system for cost-effective embedded solution. SAE Tech Paper SAE Int. https://doi.org/10.4271/2020-01-1210
31. Zhou Y, Tuzel O (2018) Voxelnet: End-to-end learning for point cloud based 3d object detection. In: 2018 IEEE conference on computer vision and pattern recognition, CVPR, pp 4490–4499. https://doi.org/10.1109/CVPR.2018.00472