Scheiner *et al. AI Perspectives* (2021) 3:6
https://doi.org/10.1186/s42467-021-00012-z

AI Perspectives

RESEARCH ARTICLE Open Access

Object detection for automotive radar point clouds – a comparison

Nicolas Scheiner[1*], Florian Kraus[1], Nils Appenrodt[1], Jürgen Dickmann[1] and Bernhard Sick[2]

Abstract

Automotive radar perception is an integral part of automated driving systems. Radar sensors benefit from their excellent robustness against adverse weather conditions such as snow, fog, or heavy rain. Despite the fact that machine-learning-based object detection is traditionally a camera-based domain, vast progress has been made for lidar sensors, and radar is also catching up. Recently, several new techniques for using machine learning algorithms towards the correct detection and classification of moving road users in automotive radar data have been introduced. However, most of them have not been compared to other methods or require next generation radar sensors which are far more advanced than current conventional automotive sensors. This article makes a thorough comparison of existing and novel radar object detection algorithms with some of the most successful candidates from the image and lidar domain. All experiments are conducted using a conventional automotive radar system. In addition to introducing all architectures, special attention is paid to the necessary point cloud preprocessing for all methods. By assessing all methods on a large and open real world data set, this evaluation provides the first representative algorithm comparison in this domain and outlines future research directions.

Keywords: Automotive radar, Object detection, Vehicular perception, Automated driving

Introduction

Automated vehicles are a major trend in the current automotive industry. Applications rank from driver assistance functions to fully autonomous driving. The vehicle perception is realized by a large sensor suite. The most prominent representatives are camera, lidar, and radar sensors. While camera and lidar have high angular resolution and dense sensor scans, automotive radar sensors have a good range discrimination. Due to their large wavelength, radar sensors are highly robust against adverse weather situations such as snow, fog, heavy rain, or direct light incidence. Moreover, they are able to estimate a relative (Doppler) velocity with a single sensor scan, or, more recently, measure polarimetric information [1, 2]. From a manufacturer's perspective, the affordability of radar sensors is another advantage. In the radar community, the most common approach to utilize Doppler values is separating the perception task for stationary and moving objects [3, 4]. In the static world, radar data can be accumulated over long time frames, because moving objects can be filtered out via their Doppler signature, thus alleviating the data sparsity problem [5–7]. Owing to their dynamic nature, moving objects require smaller time frames, typically within a few hundred milliseconds. Most detection and classification methods require an accumulation of several measurement cycles to increase the point cloud's density as depicted in Fig. 1. Nevertheless, the accumulation can be implemented in a sliding window manner, enabling updates for every single sensor cycle. So far, most research in the automotive radar domain is concerned with solely classification or instance detection. Fewer articles are available for multi-class object detection problems on dynamic road users. As the new field of automotive radar-based object detection is just emerging, many of the existing methods still use limited data sets or lack a comparison with other methods.

*Correspondence: nicolas.scheiner@daimler.com
[1]Mercedes-Benz AG, Heßbrühlstr. 21 a-d 70565, Stuttgart, Germany
Full list of author information is available at the end of the article

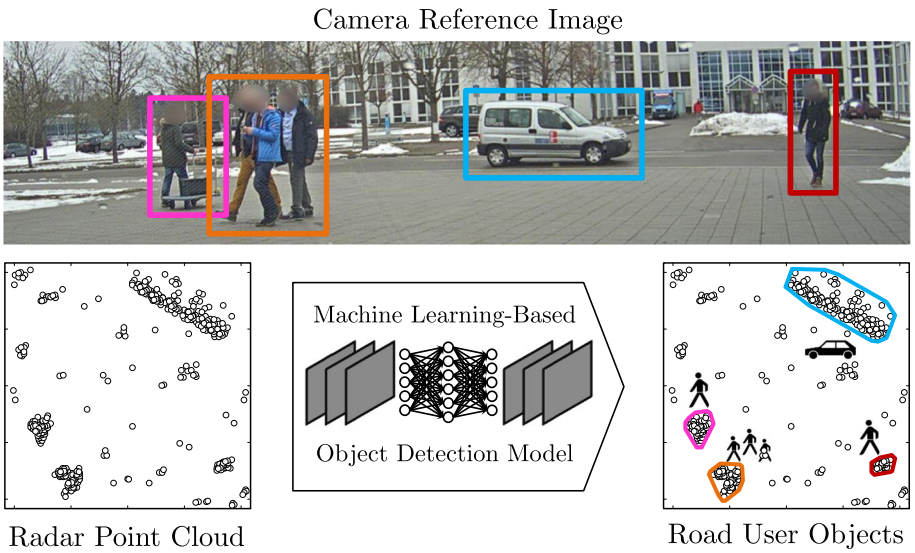


Fig. 1 The aim of this article is to detect and classify moving road users in automotive radar point clouds with help of real-time capable machine learning models. To this end, several different approaches are discussed and evaluated against each other on a large and open real world data set

In this article, the focus lies entirely on object detection, i.e., the localization and classification of an arbitrary number of moving road users. The utilized sensor system consists of conventional automotive radar sensors, i.e., no next-generation radar with high resolution, elevation, or polarimetric information is available. The data set consists of five different object classes, three vulnerable road user (VRU) and two vehicle classes. Four very different approaches plus a fifth additional combination thereof are examined for object detection: a recurrent classification ensemble, a semantic segmentation network, an image-based object detection network, and an object detector based on point clouds. The first two classification methods are extended by a clustering algorithm in order to find object instances in the point cloud. All methods are evaluated and compared against each other on a common data set. Final results indicate, advantages of the deep image detection network and the recurrent neural network ensemble over the other approaches.

Specifically, the following contributions are made:

- A representative study on five real-time capable object detector architectures is conducted.
- For comparability, a large open data set is used.
- The majority of the utilized approaches is either novel or applied for the first time to a diverse automotive radar data set.
- An in-depth overview of the preprocessing steps required to make these models accessible for automotive multi-class radar point clouds is given.
- Additional non-successful experiments are included.
- The lessons learned lead to a discussion of future research topics from a machine learning perspective.

Related work

Until recently, the majority of publications in automotive radar recognition focused on either object instance formation, e.g., clustering [8, 9], tracking [10, 11], or classification [12–16]. Object detection can be achieved, by combining instance formation and classification methods as proposed in [10, 17, 18]. This approach allows optimizing and exchanging individual components of the pipeline. However, during training, consecutive modules become less accustomed to imperfect intermediate data representations compared to end-to-end approaches.

In contrast to classifying data clusters, a second family of methods evolves around the semantic segmentation networks PointNet [19] and PointNet++ [20]. In semantic segmentation, a class label is predicted for each data point. The PointNet++ architecture is adapted to automotive radar data in [13]. The additional semantic information can, e.g., be used to train an additional classifier which aims to find objects [6]. It can also be used to regress bounding boxes as shown in [21] and [22] but requires a suitable pre-selection of patches for investigation. Since the emergence of PointNet and PointNet++, other work has focused on directly processing raw point clouds with little to no postprocessing required, such as KPConv [23], Minkowski networks [24] or various other architectures [25–28]. As such, they can be used as a replacement for PointNet(++) based semantic segmentation backbones,

albeit with adjusted (hyper)parameters. This is discussed in more detail in the Perspectives section.

Machine-learning-based object detection has made vast progress within the last few years. The two image detectors OverFeat [29] and R-CNN [30] introduced a new era of deep neural networks. Mainly due to the use of convolutional neural networks (CNN), these networks outperformed all previous approaches. Modern object detectors can be divided into two-stage and one-stage approaches. Two-stage models such as Faster R-CNN [31] create a sparse set of possible object locations before classifying those proposals. One-stage architectures combine both tasks and return a dense set of object candidates with class predictions, e.g., YOLO [32], SSD [33]. Single-stage architectures are very intriguing for automotive applications due to their computational efficiency. RetinaNet [34] further improves on those results by introducing a new "focal" loss which is especially designed for single-stage models. In parallel, also other approaches continue to evolve, e.g., YOLOv3 [35] is a more accurate successor of the YOLO family. Due to their high performance, deep CNN architectures have already made their way into automotive radar object detection. For stationary objects they can easily be applied by creating a pseudo image from radar point cloud in form of grid map, e.g., [6, 7, 36]. In [22], a YOLOv3 architecture is enabled by a grid mapping approach for moving objects, but the results on their extremely sparse data set are not encouraging. The most common approach for moving objects is to utilize a lower data level than the conventional radar point cloud, e.g., range-azimuth, range-Doppler, or azimuth-Doppler spectra [37–46] or even 3D areas from the radar data cube [47]. The advantage of these methods is that the dense 2D or 3D tensors have a format similar to images. Therefore they can be applied much easier than point clouds. However, these low level representations are usually not returned from conventional automotive radars. This is mostly due to extremely high data rates and merging problems when accumulating multiple sensor scans from different times or sensors.

Finally, CNN-based point cloud object detectors use similar ideas to image detectors. Essentially, they incorporate the process of creating as pseudo image for image detection into the model. This allows to directly train the object detector on point cloud data end-to-end without extensive preprocessing. Popular representatives of point cloud object detectors are VoxelNet [48], PointPillars [49], the structure aware SA-SSD [50], or more recently Point-Voxel-RCNN [51]. PointPillars was already applied to automotive radar in [52], but only for a non-conventional state-of-the-art radar and without any details on the implementation.

For better comparability, this article compares all four approaches, i.e., cluster classification, semantic segmentation, image-, and point-cloud-based end-to-end object detection, plus a combination of the first two methods on a common data set.

Data set

Important automotive data sets such as KITTI [53], Cityscapes [54], or EuroCity Persons [55] include only camera and at best perhaps some lidar data. In the automotive radar domain, a few new data sets were recently made public, namely nuScenes [56], Astyx [57], Oxford Radar RobotCar [58], MulRan [59], RADIATE [60], CARRADA [61], and the Zendar data set [62]. Despite the increasing number of candidates, most of them are not applicable to the given problem for various reasons.

The nuScenes data set comprises several sensors, but the output of the radar sensor is very sparse even for an automotive radar. Hence, pure radar-based experiments do not seem reasonable. The Zendar data set suffers from a similar problem. They use synthetic aperture radar processing to increase the radar's resolution by using multiple measurements from different ego-vehicle locations. Whether this approach is sufficiently robust for autonomous driving is yet to be determined. The Astyx data set has a much higher native data density. However, the limitations are its very low number of recordings, missing time information due to non-consecutive measurements, a high class imbalance, and biasing errors in the bounding box dimensions. The Oxford Radar RobotCar data set, MulRan, and RADIATE use a totally different type of rotating radar which is more commonly used on ships. A major disadvantage of their sensor is lacking Doppler information which makes it hardly comparable to any standard automotive radar. Lastly, CARRADA uses sensors on par with conventional automotive radar systems. Unfortunately, their annotations are only available for 2D projections of the radar data cube, i.e., the data set does not include the full point cloud information. This leads to the situation, in which every research team is currently developing their own proprietary data set.

The data set used in this article is one of these proprietary examples and was announced only very recently for public availability [63]. It is a large multi-class data set using a setup of four conventional 77 GHz automotive radar sensors which are all mounted in the front bumper of a test vehicle, cf. Fig. 2. It consists of annotated bird's eye view (BEV) point clouds with range, azimuth angle, amplitude, Doppler, and time information. Moreover, the ego-vehicles odometry data and some reference images are available. The point-wise labels comprise a total of six main classes[1], five object classes and one background (or static) class. The sensors are specified with resolutions in range $\Delta_r = 0.15$m, azimuthal angle $\Delta_\phi \leq 2°$, and radial

[1] Twelve classes and a mapping to six base categories are provided to mitigate class imbalance problems.

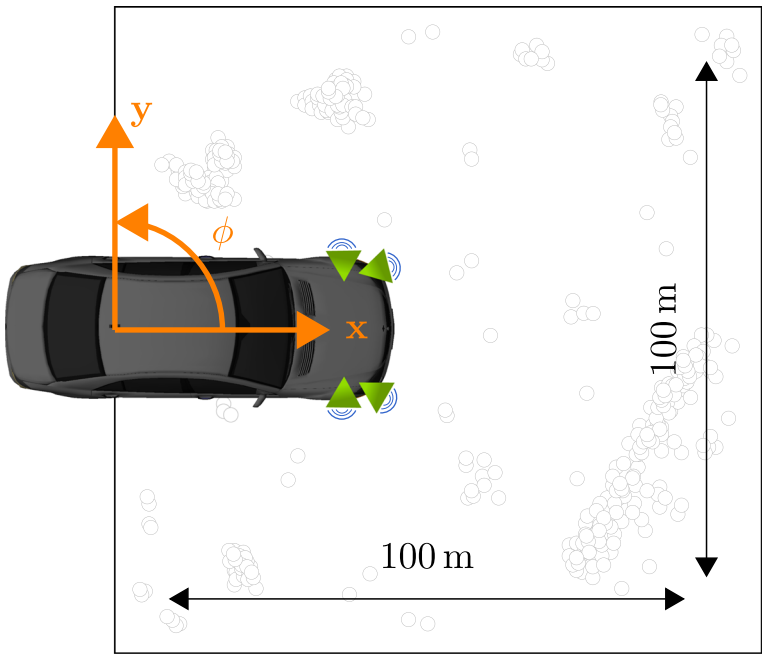


Fig. 2 Schematic sensor distribution: four sensors are located in the front bumper of the test vehicle. The origin of the coordinate system is located at the center of vehicle's rear axle. The area between 0 m $\leq x \leq$ 100 m and -50 m $\leq y \leq$ 50 m is used to create point cloud samples from the data set as shown in the image background

velocity $\Delta_{v_r} = 0.1\text{km}\,\text{s}^{-1}$. The sensor cycle time Δ_t is 60 ms. Due to data sparsity, overlapping regions in the sensors' fields of view can be superposed by simple data accumulation. In order to keep the sensors from interfering with each other, the sensor cycles are interleaved.

In order to give all evaluated algorithms a common data base, all sequences in the original data set are cropped to multiple data frames of 500 ms in time and 100 m $\times$ 100 m in space. The time frame length is a common choice among several algorithms [12, 13, 15] which were applied to the same data set prior to its publication. The spatial selection is required to avoid expensive window sweeps for grid-based detection methods (cf. "Image object detection network" and "Point-cloud-based object detection network" sections). 100 m is equal to the maximum sensor range, therefore, the crop window includes the majority of radar points. The origin of the examined area is at the middle of the vehicles rear axle, with orientation towards the driving direction of the car and symmetrical with regards to the center line of the car as depicted in Fig. 2. For this article, non-overlapping time frames are used. However, it is straight-forward, to update those time frames for every new sensor cycle. Details on this first preprocessing step are given in the Methods section. The class distribution of radar points and object instances in the cropped data samples can be found in Table 1. For the evaluation, this data set is split into roughly 64% training, 16% validation, and 20% test data. During training, only data from the train split is utilized. Validation data is used to optimize the model hyperparameters, while the results are reported based on the test data. The data splitting is done with respect to the individual sequences, i.e., all samples from one sequence are in the same split, because object instances may have similarities between different samples of the same sequence. In order to ensure a similar class distribution between all splits, a brute force approach was used to determine the best split among 10^7 sequence combinations.

Methods

In this section, all required algorithms and processing steps are described. An overview of the five most successful concepts can be gathered from Fig. 3. Additional implementational details for all methods can be found in the Appendix.

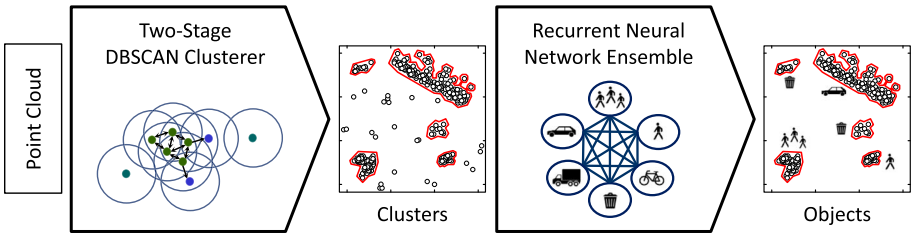
Preprocessing

As a first processing step, all range and azimuth angle values are converted to Cartesian coordinates x and y, and all

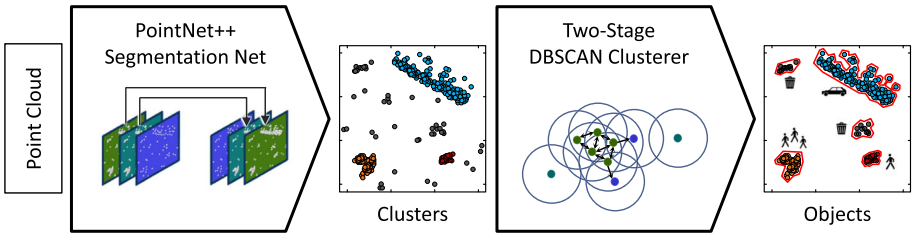
Table 1 Data set class distributions

Pedestrian	Group	Bike	Car	Truck	BG
26915	31646	8085	53721	9118	-
$5.1 \cdot 10^5$	$1.1 \cdot 10^6$	$2.7 \cdot 10^5$	$2.1 \cdot 10^6$	$9.0 \cdot 10^5$	$1.3 \cdot 10^8$

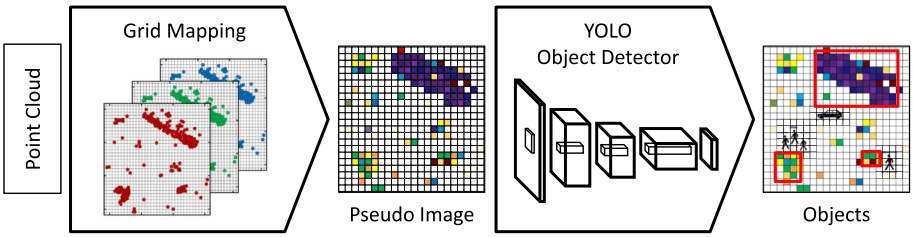
Utilized object instance (upper row) and radar detection point (lower row) distribution. The background (BG) class does not contain instances



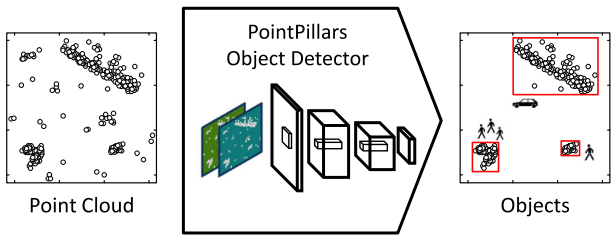
(a) DBSCAN + LSTM method.



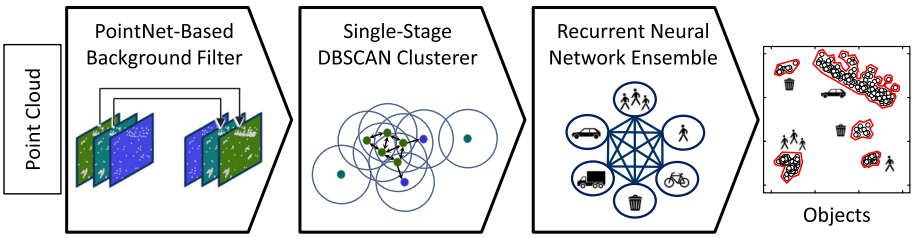
(b) PointNet++ + DBSCAN method.



(c) Grid mapping + YOLOv3 method.



(d) PointPillars method.



(e) Combined PointNet++ + LSTM method.

Fig. 3 Schematic method overview. Five main architectures are compared in this article: **a** utilizes a clustering algorithm followed by a recurrent neural network classifier. **b** makes the classification first via semantic segmentation and uses the extra information as additional input to a clusterer. **c** comprises an image-based object detector made accessible for point clouds via a grid mapping approach. **d** omits the grid mapping stage by using an object detector optimized for point clouds. Finally, **e** combines the first two methods in order to utilize the advantages of both architectures. All methods use the same point cloud as input, i.e., a cropped version of the scenario in Fig. 1. Due to space constraints, the point cloud are only displayed for the PointPillars method in **d**. Dependent on the different methods, cluster formations of boxes are returned as object predictions

radial velocity values are compensated for the ego-motion of the test vehicle. Let v_{ego} be the signed absolute velocity, $\dot{\phi}_{\text{ego}}$ the yaw rate of the ego-vehicle measured from the center of the car's rear axle, m the mounting position and rotation of the corresponding sensor in x, y, and ϕ. Then, the ego-motion compensated radial velocity $\tilde{v}_r$ can be calculated as:

$$\tilde{v}_r = v_r - \begin{pmatrix} v_{\text{ego}} + m_y \cdot \dot{\phi}_{\text{ego}} \\ m_x \cdot \dot{\phi}_{\text{ego}} \end{pmatrix}^T \cdot \begin{pmatrix} \cos(\phi + m_\phi) \\ \sin(\phi + m_\phi) \end{pmatrix}. \quad (1)$$

Where the first term of the subtrahend represents the velocity at the sensor and the second term extracts its radial component towards the compensated radar point at the azimuth angle ϕ.

To create the data samples for all methods described in this section, all data set sequences are sampled and cropped to frames of 500 ms in time and 100 m × 100 m in space. When accumulating multiple sensor cycles, it is imperative to transform all data to a common coordinate system. Therefore, all spatial coordinates are first rotated and translated from their original sensor coordinates to a car coordinate system originating from the middle of the car's rear axle. Rotation and translation values are given by the sensors' mounting positions. Second, for every sensor scan step during one 500 ms time frame, all coordinates are transformed again to the car's coordinate system at the beginning of the time frame. This time, translation and rotation are equal to translation and rotation of the ego-vehicle between the sensor scan.

Clustering and recurrent neural network classifier

As a first objection detection method, a modular approach is implemented which was previously used in [18] on a two-class detection task, cf. Fig. 3a. The point cloud is first segmented using a two-stage cluster algorithm consisting of a filtering and clustering stage based on DBSCAN [64]. The utilized two-stage clustering scheme was first introduced in [9] and is summarized in Fig. 4. In the following step, for each cluster, a set of feature vectors is created and passed to an ensemble of recurrent neural network classifiers based on long short-term memory (LSTM) cells [65].

Point Cloud Filtering Prior to the clustering stage, a data filter is applied to improve the effectiveness of the DBSCAN algorithm. In this stage, all points below some absolute velocity threshold which depends on the number of spatial neighbors N_{n} are removed. This allows for choosing wider DBSCAN parameters in the second stage without adding an excessive amount of unwanted clutter. The filtering process can be described by the following equations:

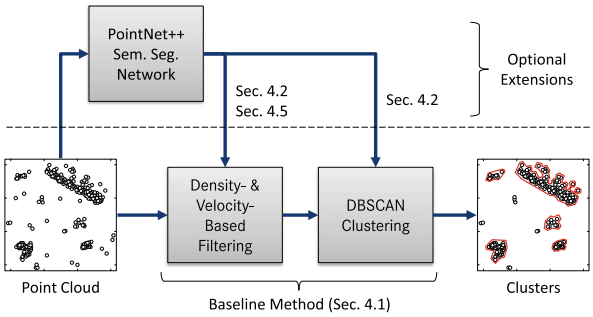


Fig. 4 Two-stage clustering scheme: A density-dependent velocity threshold is used to prefilter the point cloud. A customized DBSCAN algorithm identifies object instances, i.e. clusters, in the data. Optionally, both, the filter and the clusterer can be extended by a preceding semantic segmentation network that provides additional class information. The image indicates where each processing step is utilized among the compared methods

$$\exists i. \quad |v_r| < \boldsymbol{\eta}_{\mathbf{v_r},i} \ \wedge \ N_{\text{n}}(d_{xy}) < \mathbf{N}_i, \\ \text{with } i \in \{1, \ldots, 5\}, \mathbf{N} \in \mathbb{N}^5, \ \boldsymbol{\eta}_{\mathbf{v_r}} \in \mathbb{R}^5_{>0}, \quad (2)$$

where the velocity threshold $\boldsymbol{\eta}_{\mathbf{v_r}}$, the spatial search radius d_{xy}, and the required amount of neighbors $\mathbf{N}$ are empirically optimized parameter sets. The filter can be efficiently implemented using a tree structure, as commonly deployed for DBSCAN, or even a modified DBSCAN algorithm, hence, the term "two-stage clustering".

Clustering In the second stage, a customized DBSCAN is used to create cluster formations on the filtered point cloud. A standard DBSCAN algorithm has two tuning parameters: the minimum number of neighbors N_{min} necessary for a point to count as a core point, e.g., to spawn a new cluster and the distance threshold ϵ for any two points to be classified as neighbors. This standard algorithm is adjusted in the following ways: An additional core point criterion $|v_r| > v_{r,\text{min}}$ is used in accordance with [66]. Instead of a fixed value for N_{min}, a range-dependent variant $N_{\text{min}}(r)$ is used. It accounts for the radar's range-independent angular resolution and the resulting data density variations by requiring less neighbors at larger ranges. Details on the implementation of $N_{\text{min}}(r)$ can be found in the Appendix or in the original publication [9].

Furthermore, a new definition for the ϵ region, i.e., the multi-dimensional neighborhood distance around each point incorporates spatial distances Δx and Δy, as well as Doppler deviations Δv_r between points:

$$\sqrt{\Delta x^2 + \Delta y^2 + \epsilon_{v_r}^{-2} \cdot \Delta v_r^2} < \epsilon_{xyv_r} \ \wedge \ \Delta t < \epsilon_t. \quad (3)$$

In comparison to a single ϵ threshold, the three parameters ϵ_{v_r}, ϵ_{xyv_r} and ϵ_t allow for more fine-tuning. The time t is not included in the Euclidean distance, i.e., Δt is always required to be smaller or equal than its corresponding

threshold ϵ_t due to real-time processing constraints. All tuning parameters are adjusted using Bayesian Optimization [67] and a V_1 measure [68] optimization score. As amplitudes often have very high variations even on a single object, they are neglected completely during clustering.

Feature Extraction Once the data is clustered, a classifier decides for each cluster, whether it resembles an object instance or is part of the background class. In order to extract features for each candidate, all clusters are sampled using a sliding window of length 150 ms and a stride equal to the sensor cycle time, i.e., 60 ms. This results in 7 samples per time frame. The 150 ms time frame is an empirically determined compromise between minimum scene variation within a single sample and a maximum amount of available data points. For each of those samples, a larger vector of almost 100 handcrafted feature candidates is extracted. Features include simple statistical features, such as mean or maximum Doppler values, but also more complex ones like the area of a convex hull around the cluster. A full list of all utilized features can be found in [15]. In the following classification ensemble, each member receives an individually optimized subset of the total feature set. Subsets are estimated by applying a feature selection routine based on a guided backward elimination approach [69]. The combined feature extraction and classification process for the entire ensemble is depicted in Fig. 5.

Classification An ensemble of multiple binary classifiers which was found to outperform a single multi-class model on a similar data set in [12] is used for classification. The ensemble resembles a combined one-vs-one (OVO) and one-vs-all (OVA) approach, with $K(K+1)/2$ classifiers, where K is the total amount of classes. Note, that in order to reject a cluster proposal, it is necessary, to train the networks not only on object classes as usually done in object detection, but also on the background class. Thus, for the given six classes, the ensemble consists of 21 classifiers as indicated by the lines and circle of the classifier stage in Fig. 3a and Fig. 5. The individual classifiers each consist of a single layer of LSTM cells followed by a softmax layer. They are configured to accept exactly the seven consecutive feature vectors within one time frame. During training, class weighting is used to mitigate data imbalance effects. Training is performed on ground truth clusters. To predict the class membership of each cluster, all pairwise class posterior probabilities p_{ij} from corresponding OVO classifiers are added. Subscripts i and j denote the corresponding class identifiers for which the classifier is trained. Additionally, each OVO classifier is weighted by the sum of corresponding OVA classifier outputs q_i. This correction limits the influence of OVO classifier outputs p_{ij} for samples with class id $k \notin \{i,j\}$ [70]. The combined class posterior probabilities $\mathbf{y}(\mathbf{x})$ of the ensemble for a given feature vector $\mathbf{x}$ are then calculated as:

$$\mathbf{y}(\mathbf{x}) = \underset{i\in\{1,\dots,K\}}{\text{softmax}} \sum_{j=1,j\neq i}^{K} p_{ij}(\mathbf{x}) \cdot (q_i(\mathbf{x}) + q_j(\mathbf{x})), \tag{4}$$

where the maximum probability in $\mathbf{y}$ among non-background classes defines the identified class as well as the corresponding confidence c for this detection.

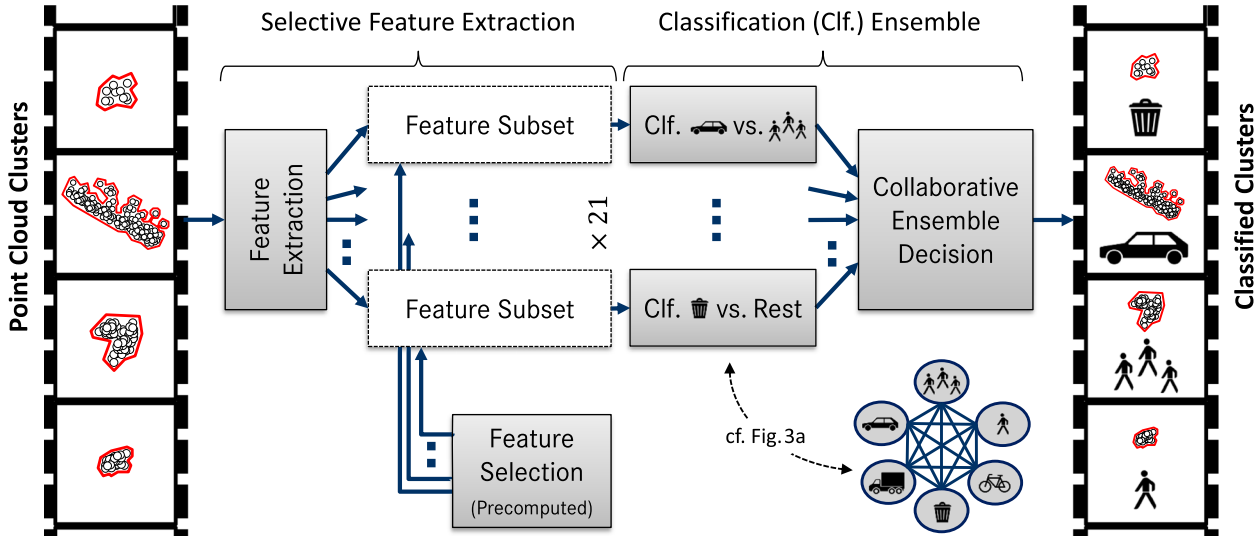


Fig. 5 Ensemble classification: A combined OVO and OVA scheme splits the decision problem into 21 binary classifiers. Each classifier obtains a custom feature set that is customized for the particular binary model. The final class decision is the product of a collaborative class decision. This process is repeated for every cluster in a point cloud

As an additional variant to this first approach, the LSTM networks in the ensemble are replaced by random forest classifiers [71] with 50 decision trees in each module (amount optimized in previous work [12]). Random forests are very fast and powerful representatives of the traditional machine learning methods. Within some limits, they allow some interpretation of their predictions. This may be advantageous when trying to explain the behavior in some exceptional cases. Instead of using samples from seven different time steps, a single feature vector is used as input to each classifier in the ensemble. Hence, the feature vector is calculated over the whole 500 ms. Due to previous classification studies [12], random forests are not expected to improve the performance. However, it is interesting whether the results differ a lot when compared to LSTM networks for object detection.

Semantic segmentation network and clustering

One of the limitations of the previous approach is its dependence on good clustering results, which are often not present. The idea behind this second approach is to improve the clustering results, by switching the order of the involved modules, i.e., classification before instance formation similar to [6]. That way, additional information, namely the class prediction, is available for the clustering algorithm as additional feature, cf. Fig. 3b. This approach requires a semantic segmentation network for point-wise class predictions.

Semantic Segmentation The utilized model is a PointNet++ architecture [19], more specifically a version specialized for automotive radar data and described in detail in [13]. It uses three multi-scale grouping modules and the same number of feature propagation modules resembling the structure of autoencoders [72]. The multi-scale grouping modules use an elaborate down-sampling method which respects local spatial context and a mini PointNet to produce features for each point of the down-sampled point cloud. Similar to deep CNNs this down-sampling process is repeated multiple (three) times. The down-sampled feature tensor is then propagated back to the point cloud by means of the feature propagation modules. This step repeatedly up-samples the point cloud, using a k-nearest neighbor approach and fully connected layers to interpolate and process the features from the coarser layer to the finer one. In addition, features are pulled in via a skip connection. From the final feature set, more fully connected layers are used to predict a class label for each original point.

The network is parameterized equally to [13] with two exceptions: PointNet++ expects a fixed number of input points, otherwise the point cloud has to be re-sampled. Instead adjusting the accumulation time to meet the size constraint, random up-sampling or point cloud splitting is used in order to use fixed time slices. Furthermore, the number of input points is optimized from 3072 to 4096 points.

Class-Sensitive Filtering and Clustering Instances formation on the classified point cloud is achieved by an adapted version of the two-stage clustering method in the Clustering and recurrent neural network classifier section. To incorporate the class labels, both, the filtering stage and the clustering stage are modified. In the filtering stage, Eq. 2 is replaced by a very simple yet effective method where a point is filtered out if:

$$|v_r| < \eta_{v_r} \ \wedge \ \arg\max \mathbf{y} = \text{background\_id}. \tag{5}$$

For the clustering stage, a new class-sensitive clustering concept is introduced. Therefore, the DBSCAN neighborhood criterion from Eq. 3 is extended by a class distance, yielding:

$$C \cdot \sqrt{\Delta x^2 + \Delta y^2 + \epsilon_{v_r}^{-2} \cdot \Delta v_r^2} < \epsilon_{xyv_r} \ \wedge \ \Delta t < \epsilon_t, \quad \text{with } C = (1 + \epsilon_c \boldsymbol{\Delta}_{\boldsymbol{c}_{ij}}) \text{ and } \Delta_{\boldsymbol{c}} \in \mathbb{R}_{\geq 0}^{K \times K}, \tag{6}$$

where $\Delta_{\boldsymbol{c}}$ is a symmetric matrix which encodes the distance between each pair of classes and ϵ_c is an additional weighting factor that is optimized. Instead of a single run, with the proposed new neighborhood criterion, multiple clustering runs are performed, one for each of the five object classes. This enables to optimize all tuning parameters towards the best setting for the corresponding class. To prevent points from object classes other than the currently regarded one to form clusters, the weights in $\Delta_{\boldsymbol{c}}$ during the clustering run for class k are set to:

$$\Delta_{\boldsymbol{c}_{ij}} = \begin{cases} 0, & \text{if } i = j = k \\ 1, & \text{if } i \neq j \wedge (i = k \vee j = k) \\ 10^6, & \text{otherwise.} \end{cases} \tag{7}$$

For parameter optimization of the clustering algorithm the PointNet++ label prediction is replaced by the ground truth label which is randomly altered using the probabilities from the class confusion matrix of the corresponding model on the validation set. Instead of this fixed setting for $\boldsymbol{\Delta}_{\mathbf{c}}$ in Eq. 7, all values could also be optimized individually. However, this would require extensive computational effort.

Note that without the class label predictions from PointNet++, it is not possible to choose class-specific optimal cluster settings as the clustering cannot be limited to the data points of a single class.

Class and Confidence Prediction Finally, for each cluster instance, a class label and a confidence value have to be selected. For the label id, a voting scheme is used to decide for the class with the most occurrences among all label predictions within the cluster. The confidence c is

then estimated by the mean posterior probability for this class within the cluster.

Image object detection network

As discussed, there are several approaches for applications of low level radar spectra to CNN-based image object detectors. In order to use point cloud data, a preprocessing step is required to arrange all data in a fixed image-like structure. In static radar processing, this is usually done via grid mapping, cf. [6, 7, 36]. The same approach can be used in dynamic processing, however, opposed to static grid maps, the data accumulation has to be limited in time to avoid object smearing. In [22], three grid maps are extracted, an amplitude map and one Doppler map for the x and y components of the measured radial velocity.

Grid Mapping For this article, a similar approach as depicted in Fig. 3c is used. The utilized CNN requires the grid size to be a multiple of 32. It is empirically optimized towards 608 × 608 cells, i.e., each cell has an edge length of ≈ 0.16 m. As in [22], three grid maps are extracted. For the first one, each grid cell is assigned the maximum amplitude value among all radar points that fall within the cell. The other two comprise the maximum and minimum signed Doppler values. This adds additional information about the Doppler distribution in each cell, in contrast to [22], where the x and y components of the radial velocities are also encoded in the cell coordinates. During grid map processing, a fourth map counts the number of accumulated radar points per cell. The map itself is not propagated to the object detector, but its information is used to copy all cells where this number exceeds an empirically set threshold, to adjacent cells. The higher the number of detections, the more neighbor cells are overwritten if they are empty. The propagation scheme is visualized in Fig. 6. When plotting the value distributions among the populated grid map cells, amplitudes resemble a normal distribution, whereas the absolute Doppler information is extremely heavy-sided towards zero. In order to ease the model's task of Doppler interpretation, a strictly monotonic increasing part of a fourth order polynomial is used to skew all Doppler values. The Doppler distribution of the training set before and after feature skewing as well as the skewing polynomial are depicted in Fig. 7. During training, grid maps are randomly rotated by multiples of 30° for data augmentation.

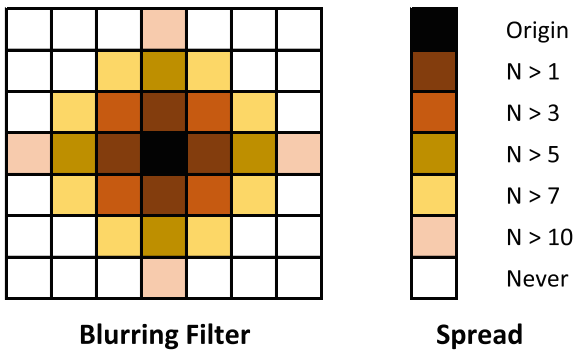


Fig. 6 Cell propagation scheme: if multiple radar points fall within a single cell, empty neighboring cells are populated with the same cell values. The color-coded blurring filter mask depends on the number of points N in the original cell

Object Detection For object detection, a YOLOv3 [35] model which poses good compromise between strong object detection accuracy and computational efficiency is applied. The implementation is based on the code provided by [73]. YOLOv3 is a one-stage object detection network. To extract features from a given image it uses Darknet-53 as a backbone. Darknet-53 is a fully convolutional network (CNN) comprised of 53 convolutional layers. The majority of those layers are arranged in residual blocks [74]. The original image is down-sampled by factors of 8, 16, and 32 to produces features at different scales. At each scale, a detection head is placed with three anchor boxes each. Hence, a total of nine anchor boxes are determined by k-means clustering on ground truth boxes. Each detection head has additional convolutional layers to further process the features with three different losses attached. First, an objectness loss $\mathcal{L}_{obj}$ to predict whether an object is present at a given bounding box location. Second, a classification loss $\mathcal{L}_{cls}$ to predict the correct class label. And third, a localization loss $\mathcal{L}_{loc}$ to regress the position of the bounding box and its size. Both, the classification and localization loss are only calculated for boxes where an object is present. The objectness loss is calculated at all locations and the total loss is the sum of all losses:

$$\mathcal{L} = \mathcal{L}_{obj} + \mathcal{L}_{cls} + \mathcal{L}_{loc}. \tag{8}$$

For this article, an extension is implemented to also regress a rotation. The angular loss $\mathcal{L}_{ang}$ is implemented as a $l1$-loss and regresses the rotation of a given bounding box. To make the loss more stable, the network is able to

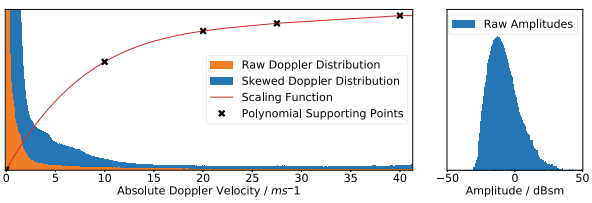


Fig. 7 Doppler value skewing: Doppler values are heavily biased towards extremely small values. Using the depicted polynomial scaling function, the distribution is widened in order to ease the feature extraction process. Amplitude values are distributed almost normal, hence no amplitude skewing is performed

either predict the ground truth rotation or its 180° equivalent. As with the $\mathcal{L}_{cls}$ and $\mathcal{L}_{loc}$, this loss is only calculated if an object is present.

At inference time, this results in multiple box predictions which are pre-selected using a confidence threshold and non-maximum suppression (NMS) [30] to discard overlapping predictions on the same location.

Point-cloud-based object detection network

The advantage of point-cloud-based object detectors is that the intermediate pseudo image representation does not need to be optimized anymore but is now part of the training routine, cf. Fig. 3d. To this end, a PointPillars [49] architecture is examined. It uses PointNets to learn the intermediate representation, which is a two-dimensional arrangement of vertical pillars in the original form. Therefore, the adapted variant for the BEV data in this article has a similar structure to the previously introduced grid maps. This structural resemblance and the fast execution time are major advantages for processing radar data with PointPillars compared other similar network types.

PointPillars consists of three main stages, a feature encoder network, a simple 2D CNN backbone, and a detection head for bounding box estimation. In the feature encoder, the pseudo image is created by augmenting a maximum of N points per pillar within a maximum of P non-empty pillars per data sample. From this augmented tensor, a miniature PointNet is used to infer a multidimensional feature space. In the CNN backbone, features are extracted at three different scales in a top-down network before being up-sampled and passed on to the detection head. The detection head uses the focal loss $\mathcal{L}_{obj}$ for object classification, another classification loss for class label assignment $\mathcal{L}_{cls}$, and a box regression loss consisting of a localization loss $\mathcal{L}_{loc}$, a size loss $\mathcal{L}_{siz}$, and an angular loss $\mathcal{L}_{ang}$ to regress a predefined set of anchor boxes to match the ground truth. The introduction of $\mathcal{L}_{cls}$ allows training a single model instead of individual ones for every class. In comparison to the original PointPillars, $\mathcal{L}_{loc}$ and $\mathcal{L}_{siz}$ are limited to the 2D parameters, i.e., the offset in x / y and length / width, respectively. Also, no directional classification loss is used since the ground truth does not contain directional object information. The total loss can be formulated as:

$$\mathcal{L} = \beta_{obj}\mathcal{L}_{obj} + \beta_{cls}\mathcal{L}_{cls} + \beta_{loc}\mathcal{L}_{loc} + \beta_{siz}\mathcal{L}_{siz} + \beta_{ang}\mathcal{L}_{ang} \quad (9)$$

The loss weights $\beta_{(\cdot)}$ are tunable hyperparameters. Besides adapting the feature encoder to accept the additional Doppler instead of height information, the maximum number of pillars and points per pillar are optimized to $N = 35$ and $P = 8000$ for a pillar edge length of 0.5 m. Notably, early experiments with a pillar edge length equal to the grid cell spacing in the YOLOv3 approach, i.e. 0.16 m, did deteriorate the results. PointPillars performs poorly when using the same nine anchor boxes with fixed orientation as in YOLOv3. Therefore, only five different-sized anchor boxes are estimated. Each is used in its original form and rotated by 90°. This results in a total of ten anchors and requires considerably less computational effort compared to rotating, i.e. doubling, the YOLOv3 anchors. For box inference, NMS is used to limit the number of similar predictions.

After initial experiments the original CNN backbone was replaced by the Darknet-53 backbone from YOLOv3 [35] to give more training capability to the network. Results of this improved PointPillars version are reported as PointPillars++.

Combined semantic segmentation and recurrent neural network classification approach

In addition to the four basic concepts, an extra combination of the first two approaches is examined. To this end, the LSTM method is extended using a preceding PointNet++ segmentation for data filtering in the clustering stage as depicted in Fig. 3e. This aims to combine the advantages of the LSTM and the PointNet++ method by using semantic segmentation to improve parts of the clustering. All optimization parameters for the cluster and classification modules are kept exactly as derived in the Clustering and recurrent neural network classifier section. As the only difference, the filtering method in Eq. 2 is replaced by the class-sensitive filter in Eq. 5.

Additional non-successful experiments

A series of further interesting model combinations was examined. As their results did not help to improve the methods beyond their individual model baselines, only their basic concepts are derived, without extensive evaluation or model parameterizations.

1) YOLO or PointPillars boxes are refined using a DBSCAN algorithm.
2) YOLO or PointPillars boxes are refined using a PointNet++ model.
3) Instead of just replacing the filter, an LSTM network is used to classify clusters originating from the PointNet++ + DBSCAN approach.
4) A semantic label prediction from PointNet++ is used as additional input feature to PointPillars.

Approaches 1) and 2) resemble the idea behind Frustum Net [20], i.e., use an object detector to identify object locations, and use a point-cloud-based method to tighten the box. Therefore, only the largest cluster (DBSCAN) or group of points with same object label (PointNet++) are kept within a predicted bounding box. As neither of the four combinations resulted in a beneficial configuration, it can be presumed that one of the strengths of the

box detections methods lies in correctly identifying object outlier points which are easily discarded by DBSCAN or PointNet++. Method 3) aims to combine the advantages of the LSTM and the PointNet++ methods by using PointNet++ to improve the clustering similar to the combined approach in the Combined semantic segmentation and recurrent neural network classification approach section. Opposed to that method, the whole class-sensitive clustering approach is utilized instead of just replacing the filtering part. However, results suggest that the LSTM network does not cope well with the so found clusters. Finally, in 4), the radar's low data density shall be counteracted by presenting the PointPillars network with an additional feature, i.e., a class label prediction from a PointNet++ architecture. At training time, this approach turns out to greatly increase the results during the first couple of epochs when compared to the base method. For longer training, however, the base methods keeps improving much longer resulting in an even better final performance. This suggests, that the extra information is beneficial at the beginning of the training process, but is replaced by the networks own classification assessment later on.

Evaluation

For evaluation several different metrics can be reported. In this section, the most common ones are introduced. Results on all of them are provided in order to increase the comparability.

Metrics

In image-based object detection, the usual way to decide if prediction matches a ground truth object is by calculating their pixel-based intersection over union (IOU) [75].

This can easily be adopted to radar point clouds by calculating the intersection and union based on radar points instead of pixels [18, 47]:

$$\text{IOU} = \frac{|\text{predicted points} \cap \text{true points}|}{|\text{predicted points} \cup \text{true points}|}. \tag{10}$$

An object instance is defined as matched if a prediction has an IOU greater or equal than some threshold. The threshold is most commonly set to 0.5. However, for a point-cloud-based IOU definition as in Eq. 10, IOU ≥ 0.5 is often a very strict condition. As an example, Fig. 8 displays a real world point cloud of a pedestrian surrounded by noise data points. The noise and the elongated object shape have the effect, that even for slight prediction variations from the ground truth, the IOU drops noticeable. For experiments with axis-aligned (rotated) bounding boxes, a matching threshold of IOU ≥ 0.1 (IOU ≥ 0.2) would be necessary in order to achieve perfect scores even for predictions equal to the ground truth. While this may be posed as a natural disadvantage of box detectors compared to other methods, it also indicates that a good detector might be neglected to seemingly bad IOU matching. Hence, in this article, all scores for IOU ≥ 0.5 and IOU ≥ 0.3 are reported.

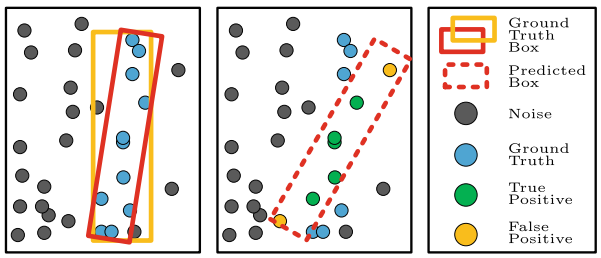


Fig. 8 IOU example of a single ground truth pedestrian surrounded by noise. On the left, the three different ground truth definitions used for different trainings are illustrated: point-wise, an axis-aligned, and a rotated box. Notably, the box variants may include noise even in the ground truth. The evaluation for all methods is solely point-based. As an example, in the middle image a slightly rotated version of the ground truth box is used as a prediction with IOU=green/(green+blue+yellow)=5/13

Once a detection is matched, if the ground truth and the prediction label are also identical, this corresponds to a true positive (*TP*). Other detections on the same ground truth object make up the false positive class (*FP*). Non-matched ground truth instances count as false negatives (*FN*) and everything else as true negatives (*TN*). The order in which detections are matched is defined by the objectness or confidence score c that is attached to every object detection output. For each class, higher confidence values are matched before lower ones.

Average precision & mean average precision

The most common object detection evaluation metrics are the Average Precision (AP) criterion for each class and the mean Average Precision (mAP) over all classes, respectively. AP uses c as control variable to sample the detector's precision $Pr(c) = TP(c)/(TP(c) + FP(c))$ at different recall levels $Re(c) = TP(c)/(TP(c) + FN(c))$:

$$\text{AP} = \frac{1}{11} \sum_{r \in \{0, 0.1, \ldots, 1\}} \max_{Re(c) \geq r} Pr(c). \tag{11}$$

For the mAP, all AP scores are macro-averaged, i.e., opposed to micro-averaging the score are calculated for each object class first, then averaged:

$$\text{mAP} = \frac{1}{\tilde{K}} \sum_{\tilde{K}} \text{AP}, \tag{12}$$

where $\tilde{K} = K - 1$ is the number of object classes.

As mAP is deemed the most important metric, it is used for all model choices in this article.

Log average miss rate

The log average miss rate (LAMR) is about the inverse metric to AP. Instead of evaluating the detectors precision, it examines its sensitivity. Therefore, it sums the miss rate $MR(c) = 1 - Re(c)$ over different levels of false positives per image (here samples) $FPPI(c) = FP/\#\text{samples}$. Using the notation from [55], LAMR is expressed as:

$$\text{LAMR} = \exp\left(\frac{1}{9}\sum_{f}\log\left(MR(\arg\max_{FPPI(c)\leq f} FPPI(c))\right)\right), \quad (13)$$

where $f \in \{10^{-2}, 10^{-1.75}, \ldots, 10^{0}\}$ denotes 9 equally logarithmically spaced *FPPI* reference points. As for AP, the LAMR is calculated for each class first and then macro-averaged. Only the mean class score is reported as mLAMR. Out of all introduced scores, the mLAMR is the only one for which lower scores correspond to better results.

F1 object score

A commonly utilized metric in radar related object detection research is the F_1, which is the harmonic mean of *Pr* and *Re*. For object class k the maximum F_1 score is:

$$F_{1,k} = \max_{c} \frac{2TP(c)}{2TP(c) + FP(c) + FN(c)}. \quad (14)$$

Again the macro-averaged F_1 score $F_{1,\text{obj}}$ according to Eq. 12 is reported.

F1 point score

Another variant of F_1 score uses a different definition for *TP*, *FP*, and *FN* based on individual point label predictions instead of class instances. For the calculation of this point-wise score, $F_{1,\text{pt}}$, all prediction labels up to a confidence c equal to the utilized level for $F_{1,\text{obj}}$ score are used. While this variant does not actually resemble an object detection metric, it is quite intuitive to understand and gives a good overview about how well the inherent semantic segmentation process was executed.

Results

The test set scores of all five main methods and their derivations are reported in Table 2. Additional ablation studies can be found in the Ablation studies section.

YOLOv3 Among all examined methods, YOLOv3 performs the best. At IOU=0.5 it leads by roughly 1% with 53.96% mAP, at IOU=0.3 the margin increases to 2%. The increased lead at IOU=0.3 is mostly caused by the high AP for the truck class (75.54%). Apparently, YOLO manages to better preserve the sometimes large extents of this class than other methods. This probably also leads to the architecture achieving the best results in mLAMR and $F_{1,\text{obj}}$ for IOU=0.3. In the future, state-of-the-art radar sensors are expected to have a similar effect on the scores as when lowering the IOU threshold. Another major advantage of the grid mapping based object detection approach that might be relevant soon, is the similarity to static radar object detection approaches. As discussed in the beginning of this article, dynamic and static objects are usually assessed separately. By combining the introduced quickly updating dynamic grid maps with the more long-term static variants, a common object detection network could benefit from having information about moving and stationary objects. Especially the dynamic object detector would get additional information about what radar points are most likely parts of the surroundings and not a slowly crossing car for example.

PointNet++ + DBSCAN + LSTM Shortly behind YOLOv3 the combined PointNet++ + DBSCAN + LSTM approach makes up the second best method in the total ranking. At IOU=0.5, it leads in mLAMR (52.06%) and $F_{1,\text{obj}}$ (59.64%), while being the second best method in mAP and for all class-averaged object detection scores at IOU=0.3. Despite, being only the second best method, the modular approach offers a variety of advantages over the YOLO end-to-end architecture. Most of all, future development can occur at several stages, i.e., better semantic segmentation, clustering, classification algorithms, or the addition of a tracker are all highly likely to further boost the performance of the approach. Also, additional fine tuning is easier, as individual components with known optimal inputs and outputs can be controlled much better, than e.g., replacing part of a YOLOv3 architecture. Moreover, both the DBSCAN and the LSTM network are already equipped with all necessary parts in order to make use of additional time frames and most likely benefit if presented with longer time sequences. Keeping next generation radar sensors in mind, DBSCAN clustering has already been shown to drastically increase its performance for less sparse radar point clouds [18]. Therefore, this method remains another contender for the future.

DBSCAN + LSTM In comparison to these two approaches, the remaining models all perform considerably worse. Pure DBSCAN + LSTM (or random forest) is inferior to the extended variant with a preceding PointNet++ in all evaluated categories. At IOU=0.3 the difference is particularly large, indicating the comparably weak performance of pure DBSCAN clustering without prior information. However, even with this conceptually very simple approach, 49.20% (43.29% for random forest) mAP at IOU=0.5 is achieved.

PointNet++ The PointNet++ method achieves more than 10% less mAP than the best two approaches. As a semantic segmentation approach, it is not surprising that it achieved the best segmentation score, i.e., $F_{1,\text{pt}}$. Also for

Table 2 Result scores for all main methods on the test set

Method	IOU=0.3									IOU=0.5								
	AP_{ped}	AP_{grp}	AP_{cyc}	AP_{car}	AP_{trk}	mAP	mLAMR	$F_{1,\text{pt}}$	$F_{1,\text{obj}}$	AP_{ped}	AP_{grp}	AP_{cyc}	AP_{car}	AP_{trk}	mAP	mLAMR	$F_{1,\text{pt}}$	$F_{1,\text{obj}}$
DBSCAN/LSTM	22.96	42.95	62.51	65.21	59.07	50.54	53.10	53.06	58.92	22.94	41.07	62.51	63.13	56.36	49.20	54.90	53.08	57.60
DBSCAN/RandomForest	20.67	37.28	53.98	59.63	56.53	45.62	59.68	49.48	52.43	20.60	35.23	53.76	55.91	50.97	43.29	61.58	49.50	51.10
PointNet++/DBSCAN	**29.42**	53.06	56.37	53.15	26.19	43.64	61.10	**54.55**	51.67	**27.47**	47.56	54.85	49.16	21.10	40.03	64.17	**54.37**	49.24
YOLOv3	28.28	**57.51**	64.87	**75.54**	**62.18**	**57.67**	**48.92**	53.04	**61.87**	26.96	**54.88**	63.68	**67.99**	56.31	**53.96**	52.61	52.86	59.46
PointPillars	15.13	32.28	46.40	61.19	47.48	40.50	65.46	47.62	45.92	14.67	30.42	45.08	54.38	39.87	36.89	68.55	47.70	43.75
PointPillars++	24.69	47.32	55.39	68.72	53.08	49.84	57.01	53.91	55.61	23.93	44.66	53.58	61.33	45.62	45.82	60.36	53.93	53.36
PointNet++/LSTM	27.28	56.28	**64.94**	68.44	60.21	55.43	50.10	54.13	60.99	27.17	51.70	**64.88**	63.74	**57.00**	52.90	**52.06**	54.12	**59.64**

For all scores except mLAMR higher means better. The methods are listed in the order of appearance in the text. The best scores are indicated in bold font

both IOU levels, it performs best among all methods in terms of AP for pedestrians. This is an interesting result, as all methods struggle the most in finding pedestrians, probably due to the latters' small shapes and number of corresponding radar points. The fact, that PointNet++ outperforms other methods for this class indicates, that the class-sensitive clustering is very effective for small VRU classes, however, for larger classes, especially the truck class, the results deteriorate. To pinpoint the reason for this shortcoming, an additional evaluation was conducted at IOU=0.5, where the AP for each method was calculated by treating all object classes as a single road user class. This effectively removes all classification errors from the AP, leaving a pure foreground object vs. background detection score. Calculating this metric for all classes, an AP of 69.21% is achieved for PointNet++, almost a 30% increase compared to the real mAP. Notably, all other methods, only benefited from the same variation by only $\approx$ 5–10%, effectively making the PointNet++ approach the best overall method under these circumstances. The high performance gain is most likely explained by the high number of object predictions in the class-sensitive clustering approach, which gives an above-average advantage if everything is classified correctly. However, it also shows, that with a little more accuracy, a semantic segmentation-based object detection approach could go a long way towards robust automotive radar detection. Nevertheless, currently it is probably best to only use the PointNet++ to supplement the cluster filtering for the LSTM method.

PointPillars Finally, the PointPillars approach in its original form is by far the worst among all models (36.89% mAP at IOU=0.5). It performs especially poorly on the pedestrian class. Normally, point-cloud-based CNN object detection networks such as PointPillars would be assumed to surpass image-based variants when trained on the same point cloud detection task. The reason for this is the expectation, that the inherent pseudo image learning of point cloud CNNs is advantageous over an explicit grid map operation as used in the YOLOv3 approach. Probably because of the extreme sparsity of automotive radar data, the network does not deliver on that potential. To alleviate this shortcoming, the original PointPillars backbone was replaced by a deeper Darknet-53 module from YOLOv3. The increased complexity is expected to extract more information from the sparse point clouds than in the original network. In fact, the new backbone lifts the results by a respectable margin of $\approx$ 9% to a mAP of 45.82% at IOU=0.5 and 49.84% at IOU=0.3. These results indicate that the general idea of end-to-end point cloud processing is valid. However, the current architecture fails to achieve performances on the same level as the YOLOv3 or the LSTM approaches. A natural advantage of PointPillars is that once the network is trained, it requires a minimal amount of preprocessing steps in order to create object detections. In turn, this reduces the total number of required hyperparameter optimization steps.

Time vs. performance evaluation

While the code for the utilized methods was not explicitly optimized for speed, the main components are quite fast. In Fig. 9, a combined speed vs. accuracy evaluation is displayed. The values reported there are based on the average inference time over the test set scenarios. For modular approaches, the individual components are timed individually and their sum is reported. The fastest methods are the standard PointPillars version (13 ms), the LSTM approach (20.5 ms) and its variant with random forest classifiers (12.1 ms). The latter two are the combination of 12 ms DBSCAN clustering time and 8.5 ms for LSTM or 0.1 ms for random forest inference. For the DBSCAN to achieve such high speeds, it is implemented in sliding window fashion, with window size equal to ϵ_t. The so achieved point cloud reduction results in a major speed improvement. Moreover, as the algorithm runs in real time, only the last processing step has to be taken into account for the time evaluation. As the method with the highest accuracy, YOLOv3 still manages to have a relatively low inference time of 32 ms compared to the remaining methods. For all examined methods, the inference time is below the sensor cycle time of 60 ms, thus processing can be achieved in real time.

Visual evaluation

Qualitative results on the base methods (LSTM, PointNet++, YOLOv3, and PointPillars) can be found in Fig. 10.

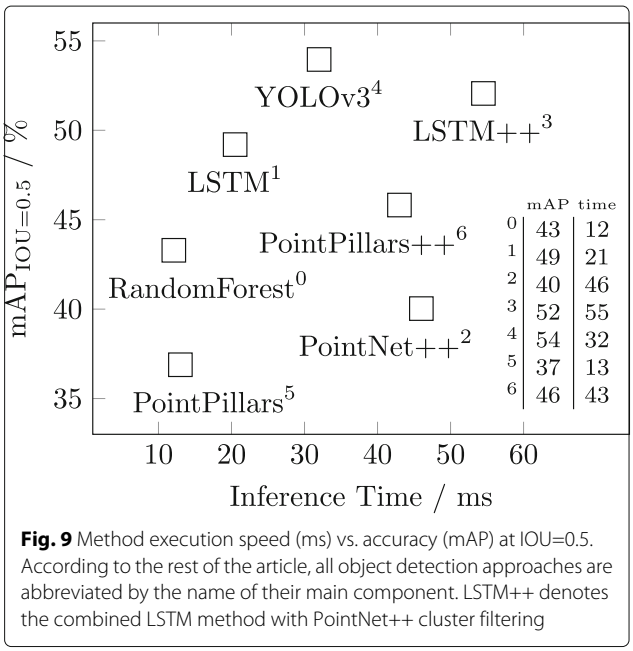


Fig. 9 Method execution speed (ms) vs. accuracy (mAP) at IOU=0.5. According to the rest of the article, all object detection approaches are abbreviated by the name of their main component. LSTM++ denotes the combined LSTM method with PointNet++ cluster filtering

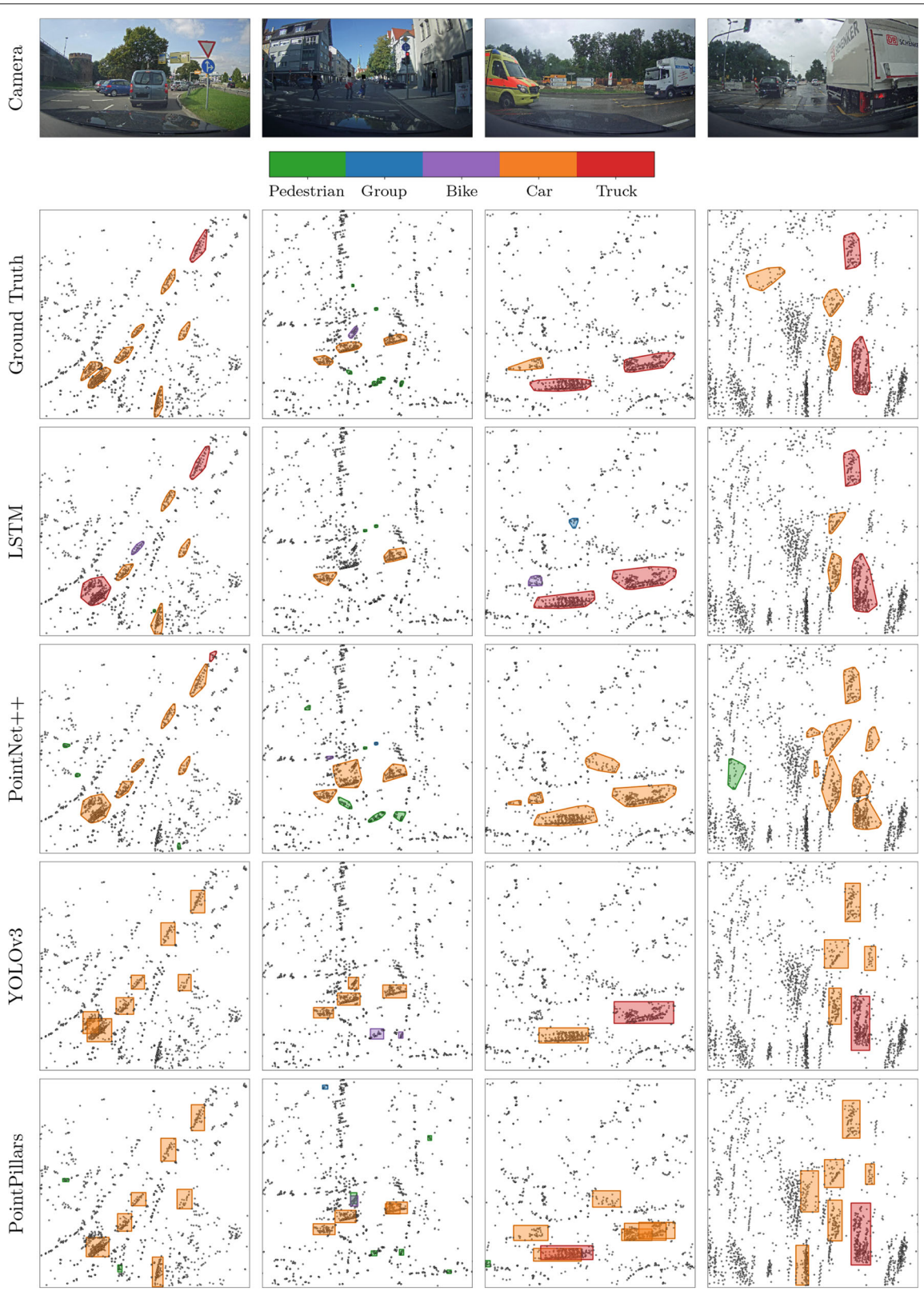


Fig. 10 Qualitative results plus camera and ground truth references for the four base methods excluding the combined approach (rows) on four scenarios (columns). Ground truth and predicted classes are color-coded

In the four columns, different scenarios are displayed. A camera image and a BEV of the radar point cloud are used as reference with the car located at the bottom middle of the BEV. The radar data is repeated in several rows. In the first view, ground truth objects are indicated as a reference. The remaining four rows show the predicted objects of the four base methods, LSTM, PointNet++, YOLOv3, and PointPillars. For each class, only detections exceeding a predefined confidence level c are displayed. The confidence level is set for each method separately, according to the level at the best found $F_{1,\text{obj}}$ score.

While for each method and scenario both positive and negative predictions can be observed, a few results shall be highlighted. In the first scenario, the YOLO approach is the only one that manages to separate the two closeby car, while only the LSTM correctly identifies the truck on top right of the image. For the second scenario, only YOLOv3 and PointPillars manage to correctly locate all three ground truth cars, but only PointNet++ finds all three pedestrians in the scene. The third scenario shows an inlet to a larger street. Despite missing the occluded car behind the emergency truck on the left, YOLO has much fewer false positives than the other approaches. Regarding the miss-classification of the emergency vehicle for a car instead of a truck, this scene rather indicates that a strict separation in exactly one of two classes can pose a problem, not only for a machine learning model, but also for a human expert who has to decide for a class label. The last scene is much more crowded with noise than the other ones. Here, the reduced number of false positive boxes of the LSTM and the YOLOv3 approach carries weight. Over all scenarios, the tendency can be observed, that PointNet++ and PointPillars tend to produce too much false positive predictions, while the LSTM approach goes in the opposite direction and rather leaves out some predictions. While this behavior may look superior to the YOLOv3 method, in fact, YOLO produces the most stable predictions, despite having little more false positives than the LSTM for the four examined scenarios.

Ablation studies

A series of ablation studies is conducted in order to help understand the influence of some method adjustments. All results can be found in Table 3.

For the LSTM method, an additional variant uses the posterior probabilities of the OVO classifier of the chosen class and the background class as confidence level. For the LSTM method with PointNet++ Clustering two variants are examined. For the first, the semantic segmentation output is only used for data filtering as also reported in the main results. The second variant uses the entire PointNet++ + DBSCAN approach to create clusters for the LSTM network. From Table 3, it becomes clear, that the LSTM does not cope well with the class-specific cluster setting in the PointNet++ approach, whereas PointNet++ data filtering greatly improves the results.

To test if the class-specific clustering approach improves the object detection accuracy in general, the PointNet++ approach is repeated with filter and cluster settings as used for the LSTM. Results indicate that class-sensitive clustering does indeed improve the results by $\approx$ 1.5% mAP, whereas the filtering is less important for the PointNet++ approach.

As mentioned above, further experiments with rotated bounding boxes are carried out for YOLO and PointPillars. Obviously, for perfect angle estimations of the network, these approaches would always be superior to the axis-aligned variants. In contrast to these expectations, the experiments clearly show that angle estimation deteriorates the results for both network types. A possible reason is that many objects appear in the radar data as elongated shapes. This is due to data accumulation over time and because, radars often just measure the object contour facing the sensors. As indicated in Fig. 8, a small offset in box rotation may, hence, result in major IOU drops. Apparently, these effects outweigh the disadvantages of purely axis-aligned predictions.

Moreover, the YOLO performance is also tested without the two described preprocessing step, i.e., cell propagation and Doppler skewing. While both methods have a small but positive impact on the detection performance, the networks converge notably faster: The best regular YOLOv3 model is found at 275k iterations. Without cell propagation, that number goes up to 300k, without Doppler scaling up to 375k, and 400k training iterations without both preprocessing steps. This supports the claim, that these processing steps are a good addition to the network.

Conclusion

In this article, an object detection task is performed on automotive radar point clouds. The aim is to identify all moving road users with new applications of existing methods. To this end, four different base approaches plus several derivations are introduced and examined on a large scale real world data set. The main concepts comprise a classification (LSTM) approach using point clusters as input instances, a semantic segmentation (PointNet++) approach, where the individual points are first classified and then segmented into instance clusters. Moreover, two end-to-end object detectors, one image-based (YOLOv3) architecture, and a point-cloud-based (PointPillars) method are evaluated. While end-to-end architectures advertise their capability to enable the network to learn all peculiarities within a data set, modular approaches enable the developers to easily adapt and enhance individual components. For example, if longer time frames than used for this article were evaluated, a straight-forward extension of the introduced clustering

Table 3 Ablation study result scores on the validation set

Method	IOU=0.3									IOU=0.5								
	AP_{ped}	AP_{grp}	AP_{cyc}	AP_{car}	AP_{trk}	mAP	mLAMR	$F_{1,\text{pt}}$	$F_{1,\text{obj}}$	AP_{ped}	AP_{grp}	AP_{cyc}	AP_{car}	AP_{trk}	mAP	mLAMR	$F_{1,\text{pt}}$	$F_{1,\text{obj}}$
PointNet++ Reference	**33.58**	**51.46**	52.15	**53.12**	38.98	**45.86**	59.94	**58.07**	52.62	**30.68**	**48.49**	51.16	**46.08**	32.16	**41.72**	**63.57**	**58.06**	**49.90**
PointNet++ Normal DBSCAN	31.78	50.17	**53.66**	50.38	29.66	43.13	62.61	57.49	50.40	30.09	44.56	**52.04**	44.25	25.60	39.31	64.99	57.45	48.22
PointNet++ Normal Filter	28.70	50.66	52.46	52.32	**44.60**	45.75	**59.66**	54.43	**53.36**	25.67	42.81	49.13	42.79	**35.03**	39.09	65.24	54.15	48.90
LSTM Reference	22.19	**48.41**	61.32	63.07	**66.98**	**52.39**	**52.67**	**55.29**	**59.61**	21.63	**45.95**	61.29	56.58	**60.03**	**49.10**	**54.69**	**54.95**	**58.15**
LSTM OVO Ranking	**22.91**	41.72	**61.94**	**65.55**	57.65	49.95	53.66	54.21	58.74	**22.86**	40.51	**61.94**	**63.71**	54.62	48.73	55.29	54.21	57.45
DBSCAN/RandomForest	19.17	41.71	49.86	57.70	59.87	45.66	59.50	50.85	53.11	19.14	39.57	49.81	51.24	58.47	43.65	61.47	50.85	51.70
LSTM++ Reference	28.90	**58.81**	58.16	**63.53**	**67.32**	**55.34**	**50.47**	**56.50**	61.46	27.55	**54.39**	58.16	60.92	**62.81**	**52.77**	**52.74**	**56.43**	59.84
LSTM++ OVO Ranking	**30.45**	56.35	**58.37**	63.43	67.25	55.17	50.83	56.03	**61.73**	**29.39**	52.41	**58.37**	**60.95**	62.57	52.74	52.87	56.08	**60.24**
LSTM PointNet++ Clustering	25.77	50.68	42.87	40.81	58.81	43.79	62.55	49.27	50.19	25.49	44.18	41.81	34.48	53.43	39.88	65.36	49.41	47.93
YOLOv3 Reference	37.04	57.77	55.89	69.12	**64.83**	**56.93**	**51.37**	**56.23**	61.13	35.04	54.66	**55.76**	**62.89**	56.63	**53.00**	**54.59**	**55.56**	58.79
YOLOv3 No Scale	29.95	58.69	56.33	69.18	64.73	55.78	51.67	54.88	60.63	31.21	**56.22**	54.73	62.22	**58.15**	52.51	55.12	54.76	58.29
YOLOv3 No Blurr	**38.00**	58.25	54.46	67.73	63.16	56.32	52.37	55.44	61.16	35.97	55.51	53.46	61.56	56.49	52.60	55.57	55.27	**58.85**
YOLOv3 No Scale No Blurr	36.70	**59.23**	55.68	68.46	61.13	56.24	52.14	54.73	61.23	**36.09**	55.30	54.13	61.61	53.68	52.16	55.52	54.82	58.80
YOLOv3 Rotated Boxes	34.44	55.71	**57.35**	**70.24**	61.89	55.93	51.42	55.06	**62.02**	31.22	50.70	54.18	61.83	55.06	50.60	56.33	54.46	58.19
PointPillars Reference	**14.82**	**27.94**	**40.50**	**57.80**	**54.87**	**39.19**	**67.10**	**48.69**	**44.90**	**14.33**	**26.25**	**39.05**	**51.35**	50.63	**36.32**	**69.43**	**48.69**	**42.91**
PointPillars Rotated Boxes	11.15	22.52	38.81	54.16	53.68	36.06	70.26	48.43	40.50	5.29	21.73	38.39	50.36	**51.93**	33.54	71.03	48.67	39.68
PointPillars++	**23.39**	**44.52**	**48.75**	**63.92**	**57.53**	**47.62**	**58.92**	**55.99**	**54.37**	**22.59**	**41.59**	**48.10**	**58.83**	**53.64**	**44.95**	**61.68**	**56.10**	**52.25**

For all scores except mLAMR higher means better. LSTM++ denotes the combined LSTM method with PointNet++ filtering during clustering. Bold font indicates the best experiment variant of each ablation study

approach would be the utilization of a tracking algorithm. Overall, the YOLOv3 architecture performs the best with a mAP of 53.96% on the test set. It has the additional advantage that the grid mapping preprocessing step, required to generate pseudo images for the object detector, is similar to the preprocessing of static radar data. Therefore, in future applications a combined model for static and dynamic objects could be possible, instead of the separation in current state-of-the-art methods.

As close second best, a modular approach consisting of a PointNet++, a DBSCAN algorithm, and an LSTM network achieves a mAP of 52.90%. While the 1% difference in mAP to YOLOv3 is not negligible, the results indicate the general validity of the modular approaches and encourage further experiments with improved clustering techniques, classifiers, semantic segmentation networks, or trackers.

As a representative of the point-cloud-based object detectors, the PointPillars network did manage to make meaningful predictions. However, with 36.89% mAP it is still far worse than other methods. In order to mitigate these shortcomings, an improved CNN backbone was used, boosting the model performance to 45.82%, outperforming the pure PointNet++ object detection approach, but still being no match for the better variants. This shows, that current object detectors for point clouds - or at least the PointPillars model - are not yet ready to fully utilize the advantage from end-to-end feature processing from very sparse automotive point clouds and take over the lead from image-based variants such as YOLOv3. However, it can also be presumed, that with constant model development, e.g., by choosing high-performance feature extraction stages, those kind of architectures will at least reach the performance level of image-based variants. At this point, their main advantage will be the increased flexibility for different sensor types and likely also an improvement in speed.

In future work, novel model architectures with even fewer data constraints such as the anchor free approaches DETR [76] or the pillars variation in [77] shall be examined. By allowing the network to avoid explicit anchor or NMS threshold definitions, these models supposedly improve the robustness against data density variations and, potentially, lead to even better results.

Perspectives

On the way towards fully autonomous vehicles, in addition to the potentials for the currently available data sets, a few additional aspect have to be considered for future algorithmic choices.

Next Generation Radar Sensors A first one is the advancement of radar sensors. It can be expected that high resolution sensors which are the current state of the art for many research projects, will eventually make it into series production vehicles. These new sensors can be superior in their resolution, but may also comprise additional measurement dimensions such as elevation [57] or polarimetric information [1]. It was already shown that an increased resolution greatly benefits radar point clustering and consequently object detection when using a combined DBSCAN and LSTM approach [18]. Current high resolution Doppler radar data sets are not sufficiently large and diverse to allow for a representative assessment of various deep neural network architectures. While it is expected that all methods will somehow profit from better resolved data, is seems likely that point-based approaches have a greater benefit from denser point clouds. At least for the current grid mapping techniques, having more radar points fall within the same grid cells should have a much smaller impact. Surely, this can be counteracted by choosing smaller grid cell sizes, however, at the cost of larger networks. Contrary, point cloud CNNs such as PointPillars already have the necessary tools to incorporate the extra information at the *same* grid size. Polarimetric sensor probably have the least benefit for methods with a preceding clusterer as the additional information is more relevant at an advanced abstraction level which is not available early in the processing chain. In comparison, PointNet++ or PointPillars can be easily extended with new features and an auxiliary polarimetric grid map [78] may serve to do the same for YOLOv3. The incorporation of elevation information on the other hand should be straight forward for all addressed strategies. Elevation bares the added potential that such point clouds are much more similar to lidar data which may allow radar to also benefit from advancements in the lidar domain.

Low-Level Data Access and Sensor Fusion A second import aspect for future autonomously driving vehicles is the question if point clouds will remain the preferred data level for the implementation of perception algorithms. Earlier in this article, several approaches using low level radar data were mentioned. Currently, the main advantage of these methods is the ordered data representation of the radar data before point cloud filtering which facilitates image-like data processing. If new hardware makes the high associated data rates easier to handle, the omission of point cloud filtering enables passing a lot more sensor information to the object detectors. The question of the optimum data level is directly associated with the choice of a data fusion approach, i.e., at what level will data from multiple radar sensor be fused with each other and, also, with other sensor modalities, e.g., video or lidar data. Different early and late fusion techniques come with their own assets and drawbacks. The methods

in this article would be part of a late fusion strategy generating independent proposals which can be fused in order to get more robust and time-continuous results [79]. Following an early fusion paradigm, complementary sensor modalities can be passed to a common machine learning model to increase its accuracy [80, 81] or even its speed by resolving computationally expensive subtasks [82]. In order to make an optimal decision about these open questions, large data sets with different data levels and sensor modalities are required.

Advances in General Purpose Point Cloud Processing

The importance of such data sets is emphasized when regarding the advances of related machine learning areas as a final third aspect for future automated driving technologies. The main focus is set to deep end-to-end models for point cloud data. Deep learning on unordered sets is a vast topic, specifically for sets representing actual geometric data such as point clouds. Many data sets are publicly available [83–86], nurturing a continuous progress in this field.

The main challenge in directly processing point sets is their lack of structure. Images consist of a regular 2D grid which facilitates processing with convolutions. Since the notion of distance still applies to point clouds, a lot of research is focused on processing neighborhoods with a local aggregation operator. Defining such an operator enables network architectures conceptually similar to those found in CNNs.

As the first such model, PointNet++ specified a local aggregation by using a multilayer perceptron. Ever since, progress has been made to define discrete convolution operators on local neighborhoods in point clouds [23–28]. Those *point convolution networks* are more closely related to conventional CNNs.

Until now, most of this work has not been adapted to radar data. Most end-to-end approaches for radar point clouds use aggregation operators based on the PointNet family, e.g. [6, 13, 21, 22]. The selection and adaptation of better suited base building blocks for radar point clouds is non-trivial and requires major effort in finding suitable (hyper)parameters. In return, it provides a great opportunity to further propel radar-based object detection. With the extension of automotive data sets, an enormous amount of research for general point cloud processing may find its way into the radar community.

Another algorithmic challenge is the formation of object instances in point clouds. In this article, an approach using a dedicated clustering algorithm is chosen to group points into instances. Qi et al. [87] use offset predictions and regress bounding boxes in an end-to-end fashion. Offset predictions are also used by [6], however, points are grouped directly by means of an instance classifier module. SGPN [88] predicts an embedding (or hash) for each point and uses a similarity or distance matrix to group points into instances. Finding the best way to represent objects in radar data could be the key to unlock the next leap in performance. Similar to image-based object detections where anchor-box-based approaches made end-to-end (single-stage) networks successful.

Appendix

Appendix In this supplementary section, implementation details are specified for the methods introduced in the Methods section.

As stated in the Clustering and recurrent neural network classifier section, the DBSCAN parameter $N_{\min}$ is replaced by a range-dependent variant. Accounting for the radar's constant angular resolution and the following data density variations at different ranges, for a given range r, the new number of minimum neighbors is:

$$N_{\min}(r) = N_{50} \cdot \left(1 + \alpha_r \cdot \left(\frac{50\,\mathrm{m}}{\mathrm{clip}(r, 25\,\mathrm{m}, 125\,\mathrm{m})} - 1\right)\right).$$

The tuning parameters N_{50} and α_r represent a baseline at 50 m and the slope of the reciprocal relation, respectively. Clipping the range at 25 m and 125 m prevents extreme values, i.e., unnecessarily high numbers at short distances or non-robust low thresholds at large ranges.

Additional model details can be found in the respective tables for the LSTM approach (Table 4), PointNet++ (Table 5), YOLOv3 (Table 6), and the PointPillars method (Table 7). The parameters for the combined model in the Combined semantic segmentation and recurrent neural network classification approach section are according to the LSTM and PointNet++ methods.

Table 4 LSTM approach

Model part	Implementation details
Clustering	Filter and modified DBSCAN with parameters: ϵ_{xyv_r}=1.04, ϵ_{v_r}=1.03, ϵ_t=0.25, $N_{\min,50}$=3.87, $v_{r,\min}$=1.00, α_r=0.99
Feature Extraction	21 individually optimized feature vectors from a feature list of 98 handcrafted features (full list in [15])
Classification Ensemble	15 OVO + 6 OVA classifiers with customized feature vectors
LSTM Classifier	Single LSTM layer with 80 cells followed by a softmax layer, learning rate 10^{-3}
Random Forest Classifier	50 trees, Gini impurity, max split $\sqrt{\text{feats.}}$, no restrictions on depth or split amount
Class Proposal	Proposal equal to clusters, ensemble score defines class decision and confidence level

Table 5 PointNet++ approach

Model part	Implementation details
Preprocessing	Random up-/down-sampling to 4096 points
Architecture	Three MSG and FP modules, parameters adopted from [13]
Clustering	Class-sensitive filtering and clustering, cf. Semantic segmentation network and clustering section
Class Proposal	Class label via cluster point voting, confidence equal to mean posterior for that class

Table 6 YOLOv3 approach

Model part	Implementation details
Grid Mapping	3 maps: max amplitude, min and max Doppler map, 608 × 608 cells (0.164 m)
GridMap Postprocessing	Use cell population to propagate highly populated cells to neighbors, skew heavy-sided Doppler values, cf. Image object detection network section
Architecture	YOLOv3 base implementation from [73], learning rate 10^{-5}, decay by a factor of 10 every 250k iterations, anchors: [(42 m, 46 m), (33 m, 17 m), (14 m, 30 m), (20 m, 5.1 m), (4.6 m, 12 m), (11 m, 12 m), (7.0 m, 5.6 m), (3.3 m, 3.3 m), (1.4 m, 1.5 m)]
Class Proposal	Top 200 NMS boxes @ IOU 0.5, proposal equal to points within predicted boxes

Table 7 PointPillars approach

Model part	Implementation details
Architecture	Max points N=35 and max pillars P=8000 (edge length 0.5 m). Loss weights adjusted to β_{obj}=2.5, β_{loc}=0.5, β_{siz}=2, β_{ang}=2 (axis-aligned β_{ang}=0), and β_{cls}=0.5. Learning rate $3 \cdot 10^{-4}$, 10 anchors: [(6.1 m, 18.0 m), (2.4 m, 15.3 m), (2.8 m, 7.6 m), (1.5 m, 4.4 m), (0.6 m, 1.5 m)]. Anchors used in original form and rotated by 90°. Pos./neg. IOU thr. 0.5 / 0.2
PointPillars++	Backbone replacement by Darknet-53
Class Proposal	Top 200 NMS boxes @ IOU 0.5, proposal equal to points within predicted boxes

Abbreviations
AP: Average Precision; BEV: bird's eye view; BG: background; CNN: convolutional neural network; FN: false negative; FP: false positive; FPPI: false positives per image; IOU: intersection over union; LAMR: log average miss rate; LSTM: long short-term memory; mLAMR: mean log average miss rate; mAP: mean Average Precision; MR: miss rate; NMS: non-maximum suppression; OVA: one-vs-all; OVO: one-vs-one; Pr: precision; Re: recall; TN: true negative; TP: true positive; VRU: vulnerable road user

Acknowledgements
Not applicable.

Funding
Not applicable.

Authors' contributions
NS wrote the manuscript and designed the majority of the experiments. FK is responsible for parts of the implementation and description of Methods in the Semantic segmentation network and clustering and Image object detection network sections. NA, JD, and BS revised the manuscript and contributed to the choice of utilized methods and evaluation metrics. All authors read and approved the final manuscript.

Availability of data and materials
The data set supporting the conclusions of this article is available in the Zenodo repository, doi: 10.5281/zenodo.4559821.

Declarations

Consent for publication
Not applicable.

Competing interests
The authors declare that they have no competing interests.

Author details
[1]Mercedes-Benz AG, Heßbrühlstr. 21 a-d 70565, Stuttgart, Germany. [2]Intelligent Embedded Systems, University of Kassel, Wilhelmshöher Allee 73, 34121 Kassel, Germany.

Received: 1 March 2021 Accepted: 22 September 2021
Published online: 16 November 2021

References

1. Tilly JF, Weishaupt F, Schumann O, Klappstein J, Dickmann J, Wanielik G (2019) Polarimetric Signatures of a Passenger Car. In: 2019 Kleinheubach Conference, IEEE. pp 1–4
2. Visentin T (2019) Polarimetric radar for automotive applications. PhD thesis, Karlsruher Institut für Technologie (KIT). https://doi.org/10.5445/KSP/1000090003
3. Dickmann J, Lombacher J, Schumann O, Scheiner N, Dehkordi SK, Giese T, Duraisamy B (2019) Radar for Autonomous Driving – Paradigm Shift from Mere Detection to Semantic Environment Understanding. In: Fahrerassistenzsysteme 2018. Springer, Wiesbaden. pp 1–17. https://doi.org/10.1007/978-3-658-23751-6
4. Zhou T, Yang M, Jiang K, Wong H, Yang D (2020) MMW Radar-Based Technologies in Autonomous Driving : A Review. Sensors 20(24). https://doi.org/10.3390/s20247283
5. Li M, Feng Z, Stolz M, Kunert M, Henze R, Küçükay F (2018) High Resolution Radar-based Occupancy Grid Mapping and Free Space Detection. pp 70–81. https://doi.org/10.5220/0006667300700081
6. Schumann O, Lombacher J, Hahn M, Wohler C, Dickmann J (2019) Scene Understanding with Automotive Radar. IEEE Trans Intell Veh 5(2). https://doi.org/10.1109/TIV.2019.2955853
7. Prophet R, Deligiannis A, Fuentes-Michel J-C, Weber I, Vossiek M (2020) Semantic segmentation on 3d occupancy grids for automotive radar. IEEE Access 8:197917–197930. https://doi.org/10.1109/ACCESS.2020.3032034
8. Kellner D, Klappstein J, Dietmayer K (2012) Grid-based DBSCAN for clustering extended objects in radar data. In: 2012 IEEE Intelligent Vehicles Symposium (IV). IEEE, Alcala de Henares. pp 365–370. https://doi.org/10.1109/IVS.2012.6232167
9. Scheiner N, Appenrodt N, Dickmann J, Sick B (2019) A Multi-Stage Clustering Framework for Automotive Radar Data. In: IEEE 22nd Intelligent Transportation Systems Conference (ITSC). IEEE, Auckland. pp 2060–2067. https://doi.org/10.1109/ITSC.2019.8916873
10. Pegoraro J, Meneghello F, Rossi M (2020) Multi-Person Continuous Tracking and Identification from mm-Wave micro-Doppler Signatures. IEEE Transactions on Geoscience and Remote Sensing. https://doi.org/10.1109/TGRS.2020.3019915
11. Tilly JF, Haag S, Schumann O, Weishaupt F, Duraisamy B, Dickmann J, Fritzsche M (2020) Detection and tracking on automotive radar data with deep learning. In: 23rd International Conference on Information Fusion (FUSION), Rustenburg. https://doi.org/10.23919/FUSION45008.2020.9190261
12. Scheiner N, Appenrodt N, Dickmann J, Sick B (2018) Radar-based Feature Design and Multiclass Classification for Road User Recognition. In: 2018 IEEE Intelligent Vehicles Symposium (IV). IEEE, Changshu. pp 779–786. https://doi.org/10.1109/IVS.2018.8500607
13. Schumann O, Hahn M, Dickmann J, Wöhler C (2018) Semantic Segmentation on Radar Point Clouds. In: 2018 21st International Conference on Information Fusion (FUSION). IEEE, Cambridge. pp 2179–2186. https://doi.org/10.23919/ICIF.2018.8455344
14. Kim S, Lee S, Doo S, Shim B (2018) Moving Target Classification in Automotive Radar Systems Using Convolutional Recurrent Neural Networks. In: 26th European Signal Processing Conference (EUSIPCO). IEEE, Rome. pp 1496–1500. https://doi.org/10.23919/EUSIPCO.2018.8553185
15. Scheiner N, Appenrodt N, Dickmann J, Sick B (2019) Radar-based Road User Classification and Novelty Detection with Recurrent Neural Network Ensembles. In: IEEE Intelligent Vehicles Symposium (IV). IEEE, Paris. pp 642–649. https://doi.org/10.1109/IVS.2019.8813773
16. Ulrich M, Gläser C, Timm F (2020) DeepReflecs : Deep Learning for Automotive Object Classification with Radar Reflections. preprint. https://arxiv.org/abs/2010.09273. Accessed 21 June 2021
17. Schubert E, Meinl F, Kunert M, Menzel W (2015) Clustering of high resolution automotive radar detections and subsequent feature extraction for classification of road users. In: 2015 16th International Radar Symposium (IRS). pp 174–179. https://doi.org/10.1109/IRS.2015.7226315
18. Scheiner N, Schumann O, Kraus F, Appenrodt N, Dickmann J, Sick B (2020) Off-the-shelf sensor vs. experimental radar – How much resolution is necessary in automotive radar classification? In: 23rd International Conference on Information Fusion (FUSION), Rustenburg. https://doi.org/10.23919/FUSION45008.2020.9190338
19. Qi CR, Yi L, Su H, Guibas LJ (2017) PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space. In: 31st International Conference on Neural Information Processing Systems (NIPS). Curran Associates Inc., Long Beach. pp 5105–5114. https://doi.org/10.5555/3295222.3295263
20. Qi CR, Liu W, Wu C, Su H, Guibas LJ (2018) Frustum PointNets for 3D Object Detection from RGB-D Data. In: IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). IEEE, Salt Lake City. pp 918–927. https://doi.org/10.1109/CVPR.2018.00102
21. Danzer A, Griebel T, Bach M, Dietmayer K (2019) 2D Car Detection in Radar Data with PointNets. In: IEEE 22nd Intelligent Transportation Systems Conference (ITSC), Auckland. pp 61–66. https://doi.org/10.1109/ITSC.2019.8917000
22. Dreher M, Ercelik E, Bänziger T, Knoll A (2020) Radar-based 2D Car Detection Using Deep Neural Networks. In: IEEE 23rd Intelligent Transportation Systems Conference (ITSC). IEEE, Rhodes. pp 3315–3322. https://doi.org/10.1109/ITSC45102.2020.9294546
23. Thomas H, Qi CR, Deschaud J-E, Marcotegui B, Goulette F, Guibas L (2019) KPConv: Flexible and Deformable Convolution for Point Clouds. In: IEEE/CVF International Conference on Computer Vision (ICCV). IEEE, Seoul. pp 6410–6419. https://doi.org/10.1109/ICCV.2019.00651
24. Choy C, Gwak J, Savarese S (2019) 4d spatio-temporal convnets: Minkowski convolutional neural networks. In: IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Long Beach. pp 3070–3079. https://doi.org/10.1109/CVPR.2019.00319
25. Atzmon M, Maron H, Lipman Y (2018) Point convolutional neural networks by extension operators. ACM Trans Graph 37(4):1–12. https://doi.org/10.1145/3197517.3201301
26. Xu Y, Fan T, Xu M, Zeng L, Qiao Y (2018) SpiderCNN: Deep Learning on Point Sets with Parameterized Convolutional Filters. In: European Conference on Computer Vision (ECCV). Springer, Munich. pp 90–105. https://doi.org/10.1007/978-3-030-01237-3_

27. Wu W, Qi Z, Fuxin L (2019) PointConv: Deep Convolutional Networks on 3D Point Clouds. In: IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). IEEE, Long Beach. pp 9613–9622. https://doi.org/10.1109/CVPR.2019.00985
28. Li Y, Bu R, Sun M, Wu W, Di X, Chen B (2018) PointCNN: Convolution On X-Transformed Points. In: Bengio S, Wallach H, Larochelle H, Grauman K, Cesa-Bianchi N, Garnett R (eds). Advances in Neural Information Processing Systems (NeurIPS), vol 31. Curran Associates, Inc., Montreal. pp 828–838. https://doi.org/10.5555/3326943.3327020
29. Sermanet P, Eigen D, Zhang X, Mathieu M, Fergus R, LeCun Y (2013) OverFeat: Integrated Recognition, Localization and Detection using Convolutional Networks. In: International Conference on Learning Representations (ICLR). CBLS, Banff. https://doi.org/10.1109/CVPR.2015.7299176
30. Girshick R, Donahue J, Darrell T, Malik J (2014) Rich feature hierarchies for accurate object detection and semantic segmentation. In: 2014 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Columbus. pp 580–587. https://doi.org/10.1109/CVPR.2014.81
31. Ren S, He K, Girshick R, Sun J (2016) Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. IEEE Trans Pattern Anal Mach Intell 39(6):1137–1149. https://doi.org/10.1109/TPAMI.2016.2577031
32. Redmon J, Divvala S, Girshick R, Farhadi A (2016) You Only Look Once: Unified, Real-Time Object Detection. In: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas. pp 779–788. https://doi.org/10.1109/CVPR.2016.91
33. Liu W, Anguelov D, Erhan D, Szegedy C, Reed S, Fu C-Y, Berg AC (2016) SSD: Single Shot MultiBox Detector. In: 2016 European Conference on Computer Vision (ECCV). Springer, Hong Kong. pp 21–37. https://doi.org/10.1007/978-3-319-46448-0
34. Lin T-Y, Goyal P, Girshick R, He K, Dollár P (2018) Focal Loss for Dense Object Detection. http://arxiv.org/abs/1708.02002. Accessed 21 June 2021
35. Redmon J, Farhadi A (2018) YOLOv3: An incremental improvement. arXiv. http://arxiv.org/abs/1804.02767. Accessed 21 June 2021
36. Lombacher J, Laudt K, Hahn M, Dickmann J, Wöhler C (2017) Semantic radar grids. In: 2017 IEEE Intelligent Vehicles Symposium (IV). IEEE, Redondo Beach. pp 1170–1175. https://doi.org/10.1109/IVS.2017.7995871
37. Sligar AP (2020) Machine Learning-Based Radar Perception for Autonomous Vehicles Using Full Physics Simulation. IEEE Access 8:51470–51476. https://doi.org/10.1109/ACCESS.2020.2977922
38. Kim W, Cho H, Kim J, Kim B, Lee S (2020) Yolo-based simultaneous target detection and classification in automotive fmcw radar systems. Sensors 20:2897. https://doi.org/10.3390/s20102897
39. Dong X, Wang P, Zhang P, Liu L (2020) Probabilistic Oriented Object Detection in Automotive Radar. In: IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW). IEEE/CVF, Seattle. https://doi.org/10.1109/CVPRW50498.2020.00059
40. Gao X, Xing G, Roy S, Liu H (2021) RAMP-CNN : A Novel Neural Network for Enhanced Automotive Radar Object Recognition. IEEE Sens J 21(4):5119–5132. https://doi.org/10.1109/jsen.2020.3036047
41. Al Hadhrami E, Al Mufti M, Taha B, Werghi N (2018) Ground Moving Radar Targets Classification Based on Spectrogram Images Using Convolutional Neural Networks. In: 19th International Radar Symposium (IRS). DGON, Bonn. https://doi.org/10.23919/IRS.2018.8447897
42. Pérez R, Schubert F, Rasshofer R, Biebl E (2019) Deep Learning Radar Object Detection and Classification for Urban Automotive Scenarios. In: Kleinheubach Conference. URSI Landesausschuss in der Bundesrepublik Deutschland e.V., Miltenberg
43. Major B, Fontijne D, Ansari A, Sukhavasi RT, Gowaikar R, Hamilton M, Lee S, Grechnik S, Subramanian S (2019) Vehicle Detection With Automotive Radar Using Deep Learning on Range-Azimuth-Doppler Tensors. In: IEEE/CVF International Conference on Computer Vision Workshop (ICCVW). IEEE/CVF, Seoul. pp 924–932. https://doi.org/10.1109/ICCVW.2019.00121
44. Brodeski D, Bilik I, Giryes R (2019) Deep Radar Detector. In: IEEE Radar Conference (RadarConf). IEEE, Boston. https://doi.org/10.1109/RADAR.2019.8835792
45. Shirakata N, Iwasa K, Yui T, Yomo H, Murata T, Sato J (2019) Object and Direction Classification Based on Range-Doppler Map of 79 GHz MIMO Radar Using a Convolutional Neural Network. In: 12th Global Symposium on Millimeter Waves (GSMM). IEEE, Sendai. https://doi.org/10.1109/GSMM.2019.8797649
46. Zhang G, Li H, Wenger F (2020) Object Detection and 3D Estimation Via an FMCW Radar Using a Fully Convolutional Network. In: IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). IEEE, Barcelona. https://doi.org/10.1109/ICASSP40776.2020.9054511
47. Palffy A, Dong J, Kooij J, Gavrila D (2020) CNN based Road User Detection using the 3D Radar Cube CNN based Road User Detection using the 3D Radar Cube. IEEE Robot Autom Lett PP. https://doi.org/10.1109/LRA.2020.2967272
48. Zhou Y, Tuzel O (2018) VoxelNet: End-to-End Learning for Point Cloud Based 3D Object Detection. In: 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). IEEE, Salt Lake City. pp 4490–4499. https://doi.org/10.1109/CVPR.2018.00472
49. Lang AH, Vora S, Caesar H, Zhou L, Yang J, Beijbom O (2019) PointPillars : Fast Encoders for Object Detection from Point Clouds. In: IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). IEEE/CVF, Long Beach. pp 12697–12705. https://doi.org/10.1109/CVPR.2019.01298
50. He C, Zeng H, Huang J, Hua X-S, Zhang L (2020) Structure aware single-stage 3d object detection from point cloud. In: IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). IEEE, Seattle. pp 11870–11879. https://doi.org/10.1109/CVPR42600.2020.01189
51. Shi S, Guo C, Jiang L, Wang Z, Shi J, Wang X, Li H (2020) Pv-rcnn: Point-voxel feature set abstraction for 3d object detection. In: IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Seattle. pp 10526–10535. https://doi.org/10.1109/CVPR42600.2020.01054
52. Scheiner N, Kraus F, Wei F, Phan B, Mannan F, Appenrodt N, Ritter W, Dickmann J, Dietmayer K, Sick B, Heide F (2020) Seeing Around Street Corners: Non-Line-of-Sight Detection and Tracking In-the-Wild Using Doppler Radar. In: IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). IEEE, Seattle. pp 2068–2077. https://doi.org/10.1109/CVPR42600.2020.00214
53. Geiger A, Lenz P, Urtasun R (2012) Are we ready for Autonomous Driving? The KITTI Vision Benchmark Suite. In: Conference on Computer Vision and Pattern Recognition (CVPR). IEEE, Providence. pp 3354–3361. https://doi.org/10.1109/CVPR.2012.6248074
54. Cordts M, Omran M, Ramos S, Rehfeld T, Enzweiler M, Benenson R, Franke U, Roth S, Schiele B (2016) The Cityscapes Dataset for Semantic Urban Scene Understanding. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR). IEEE, Las Vegas. pp 3213–3223. https://doi.org/10.1109/CVPR.2016.350
55. Braun M, Krebs S, Flohr FB, Gavrila DM (2019) The EuroCity Persons Dataset: A Novel Benchmark for Object Detection. IEEE Trans Patt Anal Mach Intell 41(8):1844–1861. https://doi.org/10.1109/TPAMI.2019.2897684
56. Caesar H, Bankiti V, Lang AH, Vora S, Liong VE, Xu Q, Krishnan A, Pan Y, Baldan G, Beijbom O (2020) nuscenes: A multimodal dataset for autonomous driving. In: IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Seattle. pp 11618–11628. https://doi.org/10.1109/CVPR42600.2020.01164
57. Meyer M, Kuschk G (2019) Automotive Radar Dataset for Deep Learning Based 3D Object Detection. In: 2019 16th European Radar Conference (EuRAD). IEEE, Paris. pp 129–132
58. Barnes D, Gadd M, Murcutt P, Newman P, Posner I (2020) The oxford radar robotcar dataset: A radar extension to the oxford robotcar dataset. In: 2020 IEEE International Conference on Robotics and Automation (ICRA), Paris. pp 6433–6438. https://doi.org/10.1109/ICRA40945.2020.9196884
59. Kim G, Park YS, Cho Y, Jeong J, Kim A (2020) Mulran: Multimodal range dataset for urban place recognition. In: IEEE International Conference on Robotics and Automation (ICRA), Paris. pp 6246–6253. https://doi.org/10.1109/ICRA40945.2020.9197298
60. Sheeny M, Pellegrin ED, Mukherjee S, Ahrabian A, Wang S, Wallace A (2020) RADIATE: A Radar Dataset for Automotive Perception. http://arxiv.org/abs/2010.09076. Accessed 21 June 2021
61. Ouaknine A, Newson A, Rebut J, Tupin F, Pérez P (2020) CARRADA Dataset: Camera and Automotive Radar with Range-Angle-Doppler Annotations. http://arxiv.org/abs/2005.01456
62. Mostajabi M, Wang CM, Ranjan D, Hsyu G (2020) High Resolution Radar Dataset for Semi-Supervised Learning of Dynamic Objects. In: 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW). pp 450–457. https://doi.org/10.1109/CVPRW50498.2020.00058
63. Schumann O, Hahn M, Scheiner N, Weishaupt F, Tilly J, Dickmann J, Wöhler C (2021) RadarScenes: A Real-World Radar Point Cloud Data Set

for Automotive Applications. Zenodo. https://doi.org/10.5281/zenodo.4559821
64. Ester M, Kriegel H-P, Sander J, Xu X (1996) A Density-based Algorithm for Discovering Clusters in Large Spatial Databases with Noise. In: 1996 2nd International Conference on Knowledge Discovery and Data Mining (KDD). AAAI Press, Portland. pp 226–231. https://doi.org/10.1016/B978-044452701-1.00067-3
65. 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
66. Schumann O, Hahn M, Dickmann J, Wöhler C (2018) Supervised Clustering for Radar Applications: On the Way to Radar Instance Segmentation. In: 2018 IEEE MTT-S International Conference on Microwaves for Intelligent Mobility (ICMIM). IEEE, Munich. https://doi.org/10.1109/ICMIM.2018.8443534
67. Mockus J (1974) On Bayesian Methods for Seeking the Extremum. In: IFIP Technical Conference. Springer, Nowosibirsk. pp 400–404. https://doi.org/10.5555/646296.687872
68. Rosenberg A, Hirschberg J (2007) V-Measure: A Conditional Entropy-Based External Cluster Evaluation Measure. In: Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning. Association for Computational Linguistics, Prague. pp 410–420. https://doi.org/10.7916/D80V8N84
69. Kohavi R, John GH (1997) Wrappers for Feature Subset Selection. Artif Intell 97(1-2):273–324. https://doi.org/10.1016/S0004-3702(97)00043-X
70. Moreira M, Mayoraz E (1998) Improved pairwise coupling classification with correcting classifiers. In: 10th European Conference on Machine Learning (ECML). Springer, Chemnitz. pp 160–171. https://doi.org/10.5555/645326.649694
71. Breiman L (2001) Random forests. Mach Learn 45(1):5–32. https://doi.org/10.1023/A:1010933404324
72. Goodfellow I, Bengio Y, Courville A (2016) Deep Learning. MIT Press, Cambridge. https://www.deeplearningbook.org
73. Kraus F, Dietmayer K (2019) Uncertainty estimation in one-stage object detection. In: IEEE 22nd Intelligent Transportation Systems Conference (ITSC). IEEE, Auckland. pp 53–60. https://doi.org/10.1109/ITSC.2019.8917494
74. He K, Zhang X, Ren S, Sun J (2016) Deep Residual Learning for Image Recognition. In: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). IEEE, Las Vegas. pp 770–778. https://doi.org/10.1109/CVPR.2016.90
75. Everingham M, Eslami SMA, Van Gool L, Williams CKI, Winn J, Zisserman A (2015) The Pascal Visual Object Classes Challenge: A Retrospective. Int J Comput Vis 111(1):98–136. https://doi.org/10.1007/s11263-014-0733-5
76. Carion N, Massa F, Synnaeve G, Usunier N, Kirillov A, Zagoruyko S (2020) End-to-End Object Detection with Transformers. In: 16th European Conference on Computer Vision (ECCV). Springer, Glasgow. pp 213–229. https://doi.org/10.1007/978-3-030-58452-8_1
77. Wang Y, Fathi A, Kundu A, Ross D, Pantofaru C, Funkhouser T, Solomon J (2020) Pillar-based Object Detection for Autonomous Driving. In: European Conference on Computer Vision (ECCV). Springer, Glasgow. pp 18–34. https://doi.org/10.1007/978-3-030-58542-6_
78. Weishaupt F, Tilly J, Dickmann J, Heberling D (2020) Polarimetric covariance gridmaps for automotive self-localization. In: 23rd International Conference on Information Fusion (FUSION), Rustenburg. https://doi.org/10.23919/FUSION45008.2020.9190231
79. Hajri H, Rahal M-C (2018) Real time lidar and radar high-level fusion for obstacle detection and tracking with evaluation on a ground truth. Int J Mech Mechatron Eng 12(8):821–827. https://doi.org/10.5281/zenodo.1474353
80. Chadwick S, Maddern W, Newman P (2019) Distant vehicle detection using radar and vision. In: International Conference on Robotics and Automation (ICRA). IEEE, Montreal. pp 8311–8317. https://doi.org/10.1109/ICRA.2019.8794312
81. Yang B, Guo R, Liang M, Casas S, Urtasun R (2020) RadarNet : Exploiting Radar for Robust Perception of Dynamic Objects. In: 16th European Conference on Computer Vision (ECCV). Springer, Glasgow. pp 496–512. https://doi.org/10.1007/978-3-030-58523-5_2
82. Nabati R, Qi H (2019) RRPN: Radar Region Proposal Network for Object Detection in Autonomous Vehicles. In: IEEE International Conference on Image Processing (ICIP). IEEE, Taipei. pp 3093–3097. https://doi.org/10.1109/ICIP.2019.8803392
83. Wu Z, Song S, Khosla A, Yu F, Zhang L, Tang X, Xiao J (2015) 3D ShapeNets: A Deep Representation for Volumetric Shapes. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR). IEEE, Boston. https://doi.org/10.1109/CVPR.2015.7298801
84. Dai A, Chang AX, Savva M, Halber M, Funkhouser T, Nießner M (2017) ScanNet: Richly-annotated 3D Reconstructions of Indoor Scenes. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR). IEEE, Honolulu. https://doi.org/10.1109/CVPR.2017.261
85. Yi L, Kim VG, Ceylan D, Shen I-C, Yan M, Su H, Lu C, Huang Q, Sheffer A, Guibas L (2016) A Scalable Active Framework for Region Annotation in 3D Shape Collections. ACM Trans Graph 35(6). https://doi.org/10.1145/2980179.2980238
86. Hackel T, Savinov N, Ladicky L, Wegner JD, Schindler K, Pollefeys M (2017) SEMANTIC3D.NET: A new large-scale point cloud classification benchmark. In: ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences, vol IV-1-W1, Hannover. pp 91–98. https://doi.org/10.5194/isprs-annals-IV-1-W1-91-2017
87. Qi CR, Litany O, He K, Guibas LJ (2019) Deep Hough Voting for 3D Object Detection in Point Clouds. In: IEEE International Converence on Computer Vision (ICCV). IEEE, Seoul. https://doi.org/10.1109/ICCV.2019.00937
88. Wang W, Yu R, Huang Q, Neumann U (2018) SGPN: Similarity Group Proposal Network for 3D Point Cloud Instance Segmentation. In: IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). IEEE, Salt Lake City. https://doi.org/10.1109/CVPR.2018.00272

Publisher's Note