applied sciences

MDPI

Article

Kernel Point Convolution LSTM Networks for Radar Point Cloud Segmentation

Felix Nobis [1,*], Felix Fent [1], Johannes Betz [2] and Markus Lienkamp [1]

[1] Institute of Automotive Technology, Technical University of Munich, 85748 Garching, Germany; felix.fent@gmx.de (F.F.); lienkamp@ftm.mw.tum.de (M.L.)
[2] mLab:Real-Time and Embedded Systems Lab, University of Pennsylvania, Philadelphia, PA 19104-6243, USA; joebetz@seas.upenn.edu
* Correspondence: nobis@ftm.mw.tum.de

Abstract: State-of-the-art 3D object detection for autonomous driving is achieved by processing lidar sensor data with deep-learning methods. However, the detection quality of the state of the art is still far from enabling safe driving in all conditions. Additional sensor modalities need to be used to increase the confidence and robustness of the overall detection result. Researchers have recently explored radar data as an additional input source for universal 3D object detection. This paper proposes artificial neural network architectures to segment sparse radar point cloud data. Segmentation is an intermediate step towards radar object detection as a complementary concept to lidar object detection. Conceptually, we adapt Kernel Point Convolution (KPConv) layers for radar data. Additionally, we introduce a long short-term memory (LSTM) variant based on KPConv layers to make use of the information content in the time dimension of radar data. This is motivated by classical radar processing, where tracking of features over time is imperative to generate confident object proposals. We benchmark several variants of the network on the public nuScenes data set against a state-of-the-art pointnet-based approach. The performance of the networks is limited by the quality of the publicly available data. The radar data and radar-label quality is of great importance to the training and evaluation of machine learning models. Therefore, the advantages and disadvantages of the available data set, regarding its radar data, are discussed in detail. The need for a radar-focused data set for object detection is expressed. We assume that higher segmentation scores should be achievable with better-quality data for all models compared, and differences between the models should manifest more clearly. To facilitate research with additional radar data, the modular code for this research will be made available to the public.

Keywords: perception; deep learning; radar segmentation; radar point cloud; object detection

check for updates

Citation: Nobis, F.; Fent, F.; Betz, J.; Lienkamp, M. Kernel Point Convolution LSTM Networks for Radar Point Cloud Segmentation. *Appl. Sci.* **2021**, *11*, 2599. https://doi.org/10.3390/app11062599

Academic Editor: Oscar Reinoso García

Received: 11 February 2021
Accepted: 11 March 2021
Published: 15 March 2021

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Copyright: © 2020 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

In everyday driving, humans perceive and react to a variety of scenarios and environment conditions. For a self-driving car to deal with such diversified situations, an extensive model of the environment is needed. For this, a variety of sensors are used to gather information about different aspects of the scene, e.g., road layout, conditions, traffic participants and traffic lights. Currently, object detection of traffic participants in 3D is most accurately performed with lidar sensor data input [1,2]. Notwithstanding, even the most accurate methods only achieve about 67% mAP (mean Average Precision) on the nuScenes data set [2] and 83% mAP on the KITTI 3D data set [1]. To gain a better understanding of the environment, object detection research is therefore pursuing two complementary approaches: Improving the detection accuracy of lidar-based algorithms; or leveraging additional sensor modalities and fusing the detection results.

This paper develops a segmentation model for radar point cloud data to complement the lidar processing. In comparison to lidar sensors, radar sensors have several advantages and disadvantages for performing object detection:

- Radar signals are affected significantly less by rain or fog than lidar [3].
- Radar sensors measure the radial velocity of surrounding objects directly.
- The spatial resolution of production radar sensors is lower than that of lidar [3].
- Current production radar sensors do not measure the elevation component of the radar returns.
- Several processing steps are necessary to obtain a radar point cloud from the raw sensor signals. These processing steps are based on additional assumptions, e.g., prioritizing moving objects, which may not lead to the desired point cloud signal in all environmental conditions.

Due to these characteristics, especially the lower resolution and the radar internal processing focus on moving objects, general object detection based on radar data is a challenging task. Figure 1 shows an example of a scene containing both radar and lidar returns. Estimating the position and the class of the shown object just from the radar returns seems more difficult than using the denser lidar data.

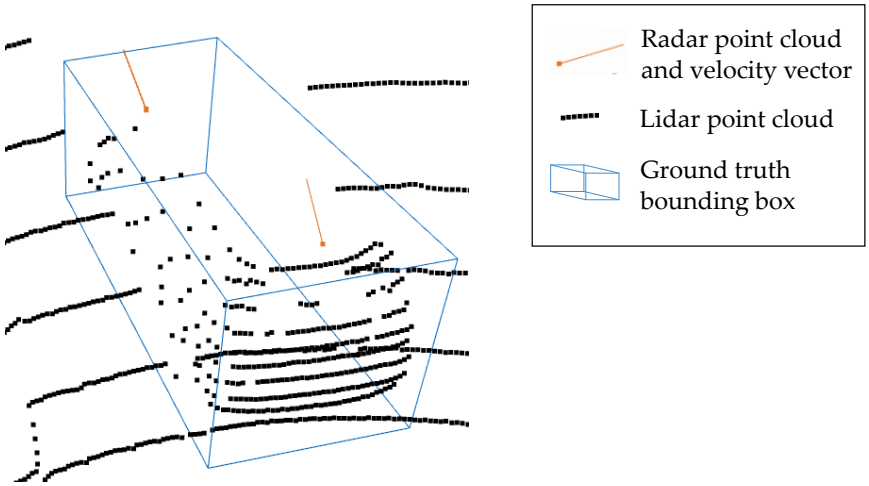


Figure 1. A vehicle detected by both radar and lidar sensors. The radar point cloud density is lower than the lidar point cloud density. Taken from nuScenes sample with the unique identifier token: 77fc24547ab34182a945eecb825b6576.

This paper compares the performance of different segmentation models on three subsets of the nuScenes data set; first on a data set distinguishing between vehicles, bikes, pedestrians and background detections; secondly on a data set only distinguishing vehicles from the background; thirdly on a data set that distinguishes moving vehicles from the background. The performance of the networks is limited by the input data quality. As data availability is one of the biggest barriers to public automotive radar research, we study the available nuScenes data set in detail and formulate requirements for an adequate radar data set for machine learning applications.

The contribution of the paper is fourfold:

- The paper adapts KPConv network architectures for radar point cloud data processing.
- The paper proposes modified LSTM network architectures to process irregular radar point cloud input data. The advantages and disadvantages of different time modelling and association techniques are discussed. However, an empirical study on public radar data does not motivate a preferred network architecture.
- The paper analyzes publicly available radar data for autonomous driving. It concludes that the data quality of the public data is not sufficient. Furthermore, it proposes which key points to consider when composing a radar data set for autonomous driving research.
- The code for this research is released to the public to make it adaptable to further use cases.

Section 2 discusses related work for lidar and radar point cloud processing. Section 3 describes the proposed models. The results are shown in Section 4 and discussed in Section 5. Our conclusions from the work are presented in Section 6.

2. Related Work

As object detection and segmentation networks often use similar backbones, we discuss both types of networks in this section. First, we give an overview of the state of the art in lidar point cloud segmentation and object detection. This motivates our choice for radar processing networks. Second, we present the state of the art in radar object detection and radar point cloud segmentation.

2.1. Lidar Point Cloud Segmentation

The task of point cloud segmentation describes the process of assigning a class label to every point of the input point cloud. As the input point cloud is a sparse, irregular 3D tensor, classic convolution neural networks cannot be directly applied to this form of input data. [4] therefore projects the data to a spherical image format to process it with 2D convolutions. The network consists of an encoder–decoder structure, to be able to generate a class label for every input pixel.

Similarly, [5] projects the point cloud data to a 2D frame. The paper augments the previously mentioned approach by taking into account a full 360° point cloud. Furthermore, it performs additional filtering for projection errors and provides an efficient implementation open source.

Qi [6] is the first to segment point cloud data without performing an intermediate feature transformation. The paper introduces the pointnet network structure, which can process an arbitrary number of input points by processing the points individually. However, their network does not consider neighboring points, as it is standard in convolutional neural networks. Their continued work [7] enhances the implementation to create local features from neighboring points while still operating on point cloud data directly. The applied operations still only process one point at a time.

Thomas [8] transfers the convolution approach from grid-based data to 3D point cloud data. The network processes a target point and its neighboring points by a defined number of kernel points analogous to 2D image convolutions. In this manner, spatial relationships can be learned directly from the input points by performing a so-called Kernel Point Convolution (KPConv).

2.2. Lidar Object Detection

3D Object detection networks estimate the position of 3D bounding boxes and associated classes in the 3D space. As they process the same input data as the segmentation networks, they can make use of the same backbone structure. Similar to point cloud segmentation, it is possible to project the point cloud data to a grid or voxel structure, or directly process the points of the point cloud to perform object detection from lidar data.

Liang [9] builds upon a spherical projected point cloud representation to extract features. In a second processing step, they project the input point cloud to a bird's-eye view (BEV) representation to estimate 3D bounding boxes.

Shi [10] presents the current state of the art in object detection according to the KITTI 3D Car leader board. They use both a projection to voxels followed by point-wise processing, to perform object detection.

VoxelNet [11] first generates point-wise features and then uses a max pooling operation to project the point features to a voxel grid structure. The object predictions are then generated by a convolutional detection head. Similarly, [12] combines pointnet and voxel-based processing to generate the object proposals.

Recurrent long short-term memory (LSTM) cells [13] are used for video processing [14] and object detection [15]. Lidar point cloud data, unlike video data, does not have a fixed structure over consecutive timeframes. When we look at a specific lidar point at a given

time, in the past time frame there might be no lidar return originating from the same location. The association of consecutive point cloud frames is therefore less intuitive than that of video data. [16] adapts LSTM cells for point cloud processing. They associate each point from the current point cloud with the past point cloud by learning a relation over the *k-nearest neighbors* and their relative distance to the current point. These association functions are used as the gate functions in the LSTM cell to create their PointLSTM to predict future point clouds from a time series.

2.3. Radar Point Cloud Processing

Radar signals can be represented in a similar data structure as lidar data. However, due to the greater sparsity and the lack of available radar data for algorithm development, the development in this area is less evolved. Many advances in the radar field are accomplished by industry research with direct access to labeled radar data. Recently, lidar processing techniques are transferred to the radar domain, taking into account the aforementioned differences between the two types of data.

Schumann [17] uses a two-step approach to classify radar objects. A DBSCAN algorithm [18] is used to cluster the radar detections. In a second step, features are generated for these clusters and classified by a random forest or a simple LSTM cell. The LSTM network outperformed the random forest approach. Difficulties with generating adequate training data for the LSTM network are mentioned. Furthermore, the drawbacks of the two-step approach and the manual corrections for the clustering algorithm are mentioned.

Another work [19] from the same group abstains from using a two-step clustering approach. Instead, they accumulate radar data over several time steps in a grid map. They create patches of 8×8 m, which are classified with a deep neural network approach. They train one-vs-all classifiers in a static environment to simplify the use case for the sparse radar data. The approach is evaluated on a proprietary data set.

Schumann [20] is the first to apply a pointnet-based approach to automotive radar data for semantic segmentation. They argue that due to the lower point cloud density grid-based discretization is not feasible for radar processing, as most grid cells would remain empty. Their approach segments different moving object classes against a static background class. It is important to note that moving objects cannot simply be distinguished from the environment by taking into account the doppler-measured velocity of the radar. Effects of an imperfect time synchronization, multi-path reflections and general noise induce non-zero doppler measurements for static locations. Furthermore, they stress the importance of taking into account the time domain for radar data processing.

A recent work by Schumann et al. [21] gives a comprehensive overview of a segmentation approach for both static and dynamic objects. They apply a convolutional encoder–decoder network on an extended grid map to classify static points. Moving points are segmented with a pointnet-based approach. The network integrates a recurrent structure by associating point features of past time steps with current point features. The time dependency is limited by the removal of old points from their memory point cloud to keep it at a fixed size. The approach is evaluated on a proprietary data set.

Palffy [22] performs radar point segmentation by using both the low-level radar cube and the processed radar point cloud data level. They evaluate their approach on the same proprietary data as [17] and show that they can set a new benchmark score by taking into account the additional data source.

Danzer [23] also uses a pointnet approach. Their network first segments patches of the environment for possible objects. Consecutively, the network processes the segmented object points to regress the bounding box dimensions for object detection. As with the other works, they evaluate their approach on a proprietary data set.

3. Methodology

Taking into account the related work above, we emphasize the importance of the following points for the development of a semantic segmentation network for radar data:

- *Data Availability*: In contrast to lidar research, where a vast amount of labeled data is publicly available, the access to radar data is more restricted. Accurately labeled data is the basis for any successful supervised learning model, which means that reduced data availability could be one reason for the smaller amount of research work in this area. A more detailed view of radar data available to the public is given in Section 5.2.
- *Data Level*: Due to the sensor principle of the radar, a variety of intermediate processing stages occur before the raw receive signal is converted into object proposals.
  The unprocessed low-level raw antenna signals theoretically contain the most information; however, there are no methods of labeling data on this abstraction level for object categories. Additionally, depending on the antenna characteristics and configuration, the data would differ a lot for different radar types, which would require a specific learning approach for different sensor hardware designs. Consequently, to the best of our knowledge, no one-step learning approach exists for classifying objects directly on the raw antenna signals.
  One can perform object detection at the radar cube level or on any of its 2D projections, e.g., range-azimuth plot. However, data labeling of this 3D representation (2D location + velocity) would still be a tedious task. [22] performs the labeling by extracting data from the cube at manually labeled point target locations. However, in this way the label quality can only approximate the ground-truth, as the point target might not include all raw signals that originate from an object. Furthermore, the point targets only represent a compression of the radar signal, which means that the point signal cannot be re-projected precisely on the cube-level signal. To our knowledge, no full 3D cube-level radar data set has been released to the public. However, due to the high informativeness and the similarity to image data, which is heavily processed with learning-based techniques, the cube-level data is an interesting use case for future deep-learning research on radar data.
  Point level data is the most explored data level for deep learning on radar data. It represents a compromise between data informativeness and data amount. Due to the similarity to lidar point cloud data, algorithms can be transferred from the lidar domain to the radar domain. This work operates on point level radar data. We benchmark our models on the public nuScenes data set [2], which includes labeled point cloud radar data.
- *Data Sparsity*: For lidar data processing, both grid and point-based approaches, are used in the state of the art. Similar to [20], we argue that for radar data, a point-based approach is more feasible than the intermediate grid representation as most of the grid cells would remain empty. For fusion approaches of radar data with denser data such as images or lidar both grid-based and point-based approaches [24–26], might be suitable. This work presents a point-based approach. To the best of our knowledge, we are the first to adapt the KPConv architecture for radar point cloud data.
- *Time Dependency*: Due to the low spatial resolution of radar sensors, the time domain plays an important role in radar processing. Current production radar systems track point level detections over several timeframes before recognizing a track as an object. This decreases the number of false positive object detections due to clutter. These false positives would otherwise be harmful to driver assistant systems such as adaptive cruise control (ACC) or emergency brake assist. In the literature, approaches that use the time domain in learning models such as simple LSTM cells or memory point clouds have proven successful for radar data processing. This paper integrates KPConv layers into an LSTM cell, which we call KPConv-based LSTM (KPLSTM). In this way, we can encode the time dependency for points individually. Additionally, we combine KPConv encoding with LSTM and ConvLSTM cells [27] in the latent space to integrate the time domain in the model on a global feature level.
- *Moving Object Recognition*: In addition to the time dependency, production applications only react to moving objects, as the amount of clutter among the static points causes

additional challenges for the processing. It is desired to bring radar processing to a level of maturity to detect static objects just as with the lidar sensor. However, the research above shows that the focus is still mainly on the moving object case. Even in the simplified moving objects scenario, the detections are only reliable enough for simplified use cases, such as vehicle following in the ACC case. In this work, we evaluate our network for both static and dynamic cases and show the disadvantages of the radar for the complex static scenarios.

- *One-vs-all Classification*: Due to the difficulties in creating reliant object detections from radar data, one compensation strategy is the simplification of the scenarios. In the literature, one-vs-all classification are an efficient way to simplify the training use case, while keeping a relevant task in focus. This paper shows the results for different training configurations and their impact on the segmentation performance.

Radar *Data Availability* is discussed in Section 5.2, The *Data Level* is given as only point level data is available. *Data Sparsity* and *Time Dependency* are major considerations for the design of the KPConv-LSTM networks presented, which are discussed in Section 3.1. The evaluation of focusing on *Moving Object Recognition* and *One-vs-All Classification* is shown in Section 4.

3.1. Model

The proposed radar models are inspired by lidar processing and general LSTM networks from literature. The models are implemented in a modular fashion so that different architectures, e.g., for the encoder or decoder part of the network, can be replaced and combined for greater model flexibility. For additional details and analysis, we refer to the master thesis of Fent [28] as the main contribution to the implementation. In the following, we recapitulate the ideas of the KPConv layer and describe our LSTM extensions, including a KPConv-based LSTM (KPLSTM) cell. The proposed models can process radar input points clouds of varying size. The KPLSTM cell associates point clouds of varying sizes over consecutive time steps. In contrast to existing LSTM cells, we incorporate the time dimension as early as in the encoding part of the network.

3.2. KPConv with LSTM Cells in Latent Space

The KPConv layer is inspired by the KPConv model introduced by [8]. The idea behind the Kernel Point Convolutions is to model the spatial relationships in a point cloud directly. Sliding a standard convolutional kernel over an unordered space is not feasible. The KPConv layer applies convolutional weights at Kernel Point locations with a distance factor to neighboring points of the input point cloud. The number of kernel points K is fixed, while the number of neighboring points N_x and the respective association function h_{ij} varies with the number of points in the local neighborhood. The considered neighbor points lie within a specified radius r from the center point of the convolution. The output feature dimension d_{out} for every center point is determined by the number of KPConv filters applied. Here, the KPConv layer follows the same principle as grid-based convolutions. Figure 2 shows a graphical representation for the processing of a center point x.

Due to the sparsity of the radar data, the maximum radius for neighbor points that affect the feature output of the current center point in the first KPConv layer is set to 8 m. In the original KPConv implementation [8] for lidar data, this radius is set to a maximum of 0.15 m for outdoor scenarios. The larger radius empirically provided the best results on our data set. From a theoretical perspective, the greater sparsity of the radar data motivates a greater radius of influence to include sufficient information of the surroundings. In this radius, all radar reflections of a passenger vehicle could influence all other reflections from the same object. For larger objects, such as trucks and buses, this is not always the case. Nonetheless, an even greater radius did not result in better detection scores, as at the same time more clutter information could be associated in the neighborhood. Figure 1 shows that radar points do not necessarily originate from the visible edges of an object.

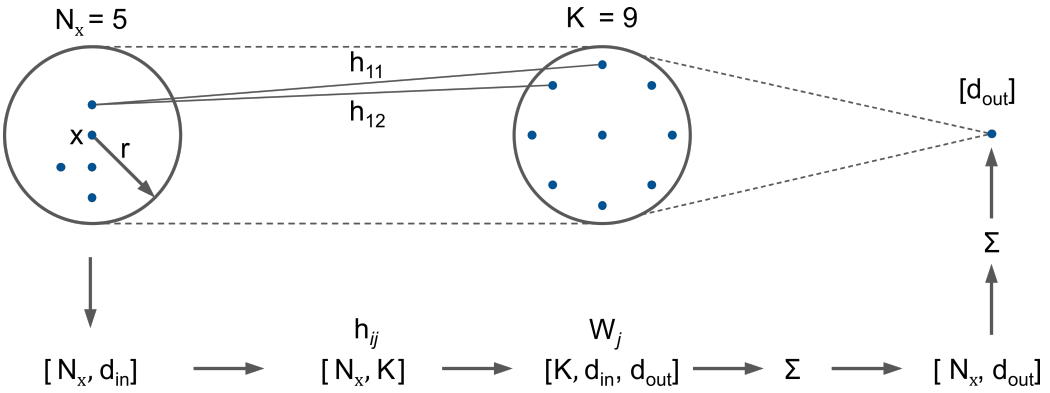


Figure 2. Processing of the center point x with d_{out} kernel point filters. Based on [8].

Figure 3 shows the high-level model structure of a KPConv model with a ConvLSTM center layer (KPConv-CLSTM). The inputs to the network are the n input radar points consisting of the last three radar time steps with five feature dimensions $feat$: Location x, y, z; radial velocity components v_x, v_y; and the radar signal strength, called radar cross section RCS. Following the original implementation, each *KPConv ResNet Block* consists of an MLP layer, a KPConv layer and another MLP layer. The skip link branch consists of an MLP layer to adapt the feature dimension before adding it to the processed KPConv branch. The number of points is compressed with a farthest point sampling method. Once the input point cloud has been encoded, convolutional LSTM cells [27] are applied to the features in the latent space representing the entire input point cloud. The decoding or upsampling of the point cloud is performed by a combination of three-nearest neighbors upsampling and MLP layers for feature generation. When features have been obtained for every input point, three MLP layers serve as a classification head. The upsampling and classification head is inspired by [20]. A class-weighted focal loss function [29] is used for training to handle class imbalances, e.g., in the *Moving Vehicle Data Set* used below over 97% of the points belong to the background class.

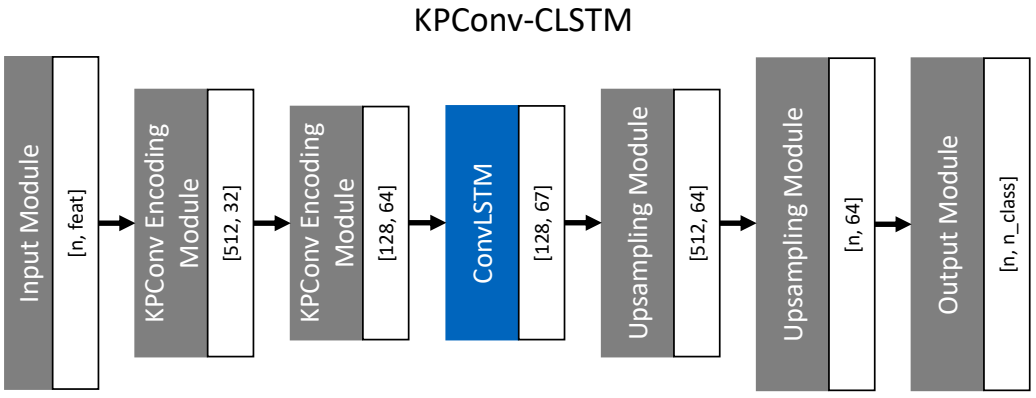


Figure 3. High-level structure of our KPConv model with ConvLSTM cells in the latent space. The blue color shows layers that consider time dependencies.

Due to the modularity of the model, we can replace the ConvLSTM layer in the latent space with a standard LSTM cell or omit the time dependency altogether for our evaluation in Section 4. Additional details of the implementation can be found directly in the provided configurations in the repository released with this publication.

3.3. KPLSTM

From an architecture point of view, the KPConv-CLSTM comes with the drawback that the time dependency is only applied on the global feature of the point cloud in the latent space. A direct application of the LSTM cells to the input points would not be feasible

due to the irregular structure and size of the input point cloud over several time steps. Schumann [21] associates points of the current radar point cloud with neighboring points from a memory point cloud by using the grouping method of PointNet++, using only the current point cloud as center points. Similar to [16], we propose a new variant of the LSTM cell in such a way that it can process and associate point cloud data directly. Instead of pointnet-based processing, we use KPConv kernel gates for the KPLSTM cell. To align the dimension of the past and present point clouds, we explore different sampling strategies from nearest neighbor sampling to learnable sampling via the KPConv weights. For the use case of the paper, the nearest neighbor sampling proved most reliant. Figure 4 shows a comparison of a standard LSTM cell and the KPLSTM cell.

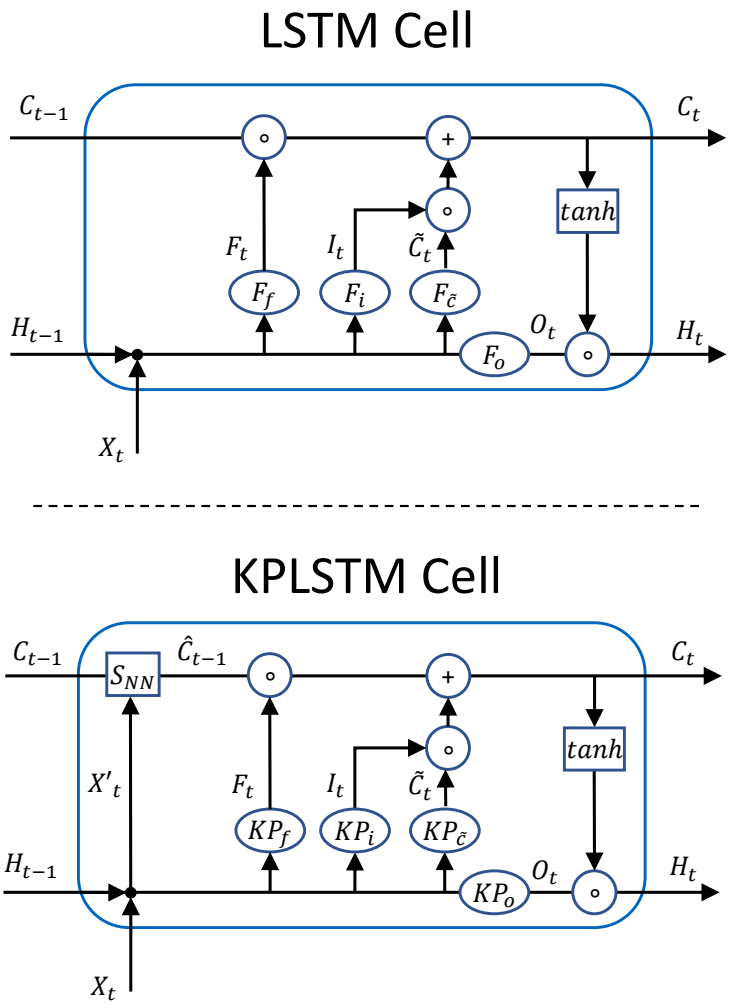


Figure 4. Comparison of standard LSTM cell and KPLSTM cell.

The inputs to the LSTM cells are the old cell state C_{t-1}, the old hidden state H_{t-1} and the current point cloud X_t. The output of the cells are the current cell state C_t and the current hidden state H_t. For the input data, the old hidden state and current point cloud are concatenated. This structure is the same as in the standard LSTM cell. Instead of fully connected layers, KPConv operations KP_x are used to model the input gate I_t, output gate O_t, forget gate F_t and internal cell state $\tilde{C}_t$. The subscript x denotes the respective weights. The KP_x layers perform feature generation and point cloud association of the old and current coordinates at the same time. Additionally, the old cell state C_{t-1} needs to be mapped to the new point coordinates. As we do not want to apply the features of the new point cloud but only propagate the old features to new coordinates, we take the current point coordinates without the point features X'_t and use them as the center points for the

nearest neighbor sampling S_{NN} to obtain the intermediate cell state $\hat{C}_{t-1}$. The remaining structure is analogous to the standard LSTM cell. The symbol $\circ$ denotes the element-wise product. We remove old features from the cell state via the forget gate and add new ones via the input gate. The equations describing the KPLSTM cell are as follows:

$$\hat{C}_{t-1} = S_{NN}(X'_t, C_{t-1}), \tag{1}$$
$$I_t = \sigma(KP_i(X_t, H_{t-1})), \tag{2}$$
$$F_t = \sigma(KP_f(X_t, H_{t-1})), \tag{3}$$
$$O_t = \sigma(KP_o(X_t, H_{t-1})), \tag{4}$$
$$\tilde{C}_t = \tanh(KP_{\tilde{c}}(X_t, H_{t-1})), \tag{5}$$
$$C_t = F_t \circ \hat{C}_{t-1} + I_t \circ \tilde{C}_t, \tag{6}$$
$$H_t = O_t \circ \tanh(C_t) \tag{7}$$

Figure 5 shows the overall KPLSTM network architecture. The encoding is performed in two consecutive blocks. Each block comprises a KPConv layer to downsample the point cloud, a KPLSTM cell and another KPConv layer for feature generation. The upsampling is done in the same manner as in the KPConv-CLSTM. It is worth mentioning that the time dependency is already introduced in the network encoding layers, making an intermediate global feature LSTM model obsolete. With this, we capture local structure over time instead of only being able to keep track of the features globally.

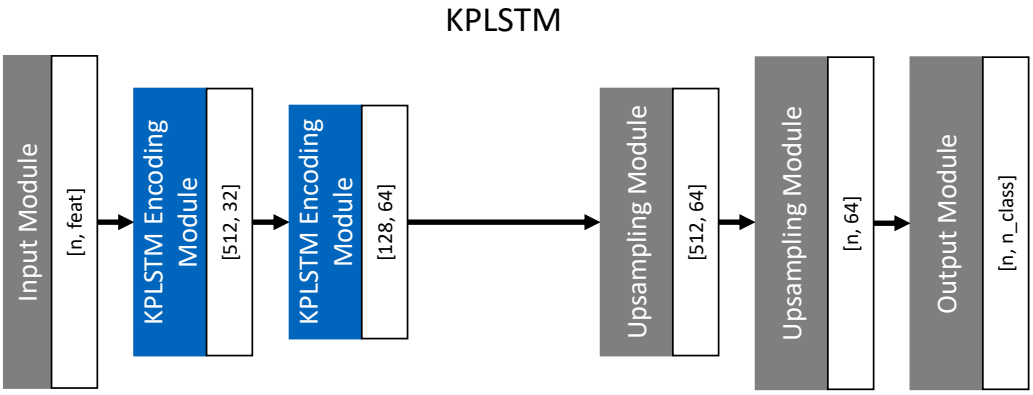


Figure 5. High-level structure of the KPLSTM model. The time dependency is already considered in the encoding stage as indicated by the blue color-coding.

3.4. Data Processing

The nuScenes data is labeled using lidar data. Radar detections are not time synchronized with the labels. Due to the time delay, bounding box labels do not always coincide with the radar detections. Further errors can be introduced through slight angular miscalibrations of the radar sensors, which has a particular impact on far-range detections. The transformation from the radar and lidar frame to the car coordinate system can be another source of spatial offset between ground-truth (GT) bounding boxes and radar data. To account for spatio-temporal calibration errors, nuScenes added the so-called *wlh*-factor to increase the bounding box size. For small bounding boxes, e.g., pedestrians, increasing the bounding box size by 50% might be a reasonable factor, whereas this would include too much environment information in the case of vehicles or buses. Similarly, the *wlh*-factor scales the tolerance for the length and width of vehicles differently, as usually the length of a vehicle is larger than its width. We propose an absolute *wlh*-tolerance in meters that is equally suited for all object categories. Figure 6 shows the misalignment of bounding boxes and the effect of the *wlh*-factor vs the *wlh*-tolerance. Although choosing a higher *wlh*-factor could also include respective points in the bounding box in Figure 6b, this would; however, increase the tolerance in longitudinal direction too much.

Due to the sparsity of radar data, we process three consecutive time steps as a single point cloud. As these past time steps are not available for the first sample of each scene of the public data set, we omit the first sample, both for training and evaluation.

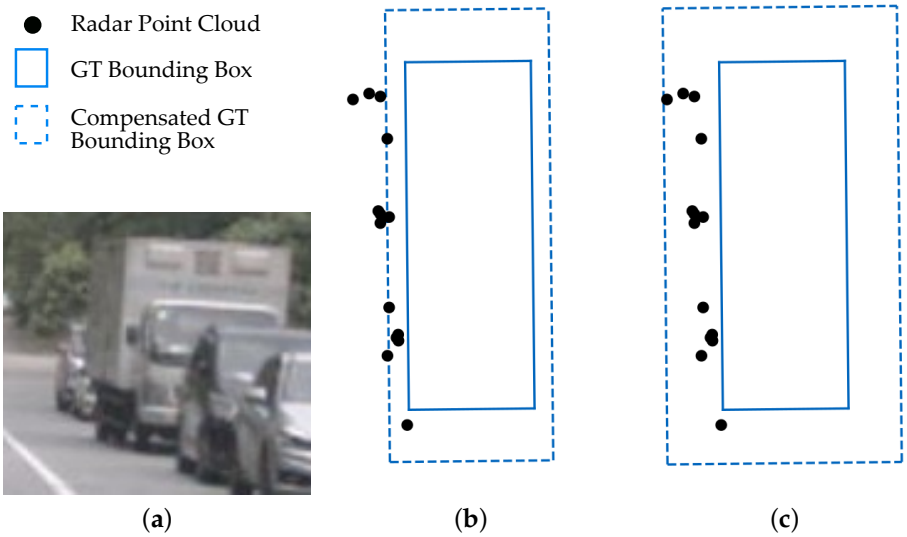


Figure 6. Compensating the misalignment of the manually labeled ground-truth bounding box with a *wlh*-tolerance creates a more precise ground-truth for the network training. (**a**) The truck of interest in camera view. (**b**) BEV: The *wlh*-factor = 1.3 includes only 5 points in the compensated GT bounding box. (**c**) BEV: The *wlh*-tolerance = 1.2 includes all 14 points in the compensated GT bounding box. NuScenes sample token: a3c6db2751a54c8590c4d8241e01ac8c.

3.5. Training

We train the segmentation networks on the official training split of the public nuScenes data set. We test the performance of the models on the official validation split as nuScenes does not provide segmentation ground-truth for their test set. We do not perform hyperparameter optimization on the validation split. During the training process, we batch several scenes for one training step. We keep the order of the samples inside each scene, to be able to learn the time dependency of the radar data.

4. Evaluation

The model performance is measured with the macro-averaged F1-score as in the works of [20–22]. However, none of the related work benchmarked their implementations on a public radar data set. To compare our approach to the literature, we implement a PointNet++ approach in the style of [20]. We use our implementation of the PointNet++ approach as the state-of-the-art baseline as there are no radar pointnet implementations publicly available.

4.1. Traffic Data Set

First, the models are trained on a data set distinguishing between the classes: vehicle, cycle, pedestrian and background. The classes consist of moving and static objects alike. Figure 7 shows the confusion matrices for three different class weight configurations. The *Veh Weights* configuration shows a strong bias towards the vehicle class, whereas almost no points are classified as pedestrians or bikes. For the *Lower Veh Weights* configuration the class weight on the vehicle class is halved. We see a better fit for the background class in this way. Additional equilibration between the class weights lead to the result of the *Distributed Weights* confusion matrix. All classes are associated while the overall quality stays low. The radar data is too sparse to reliably distinguish between four classes.

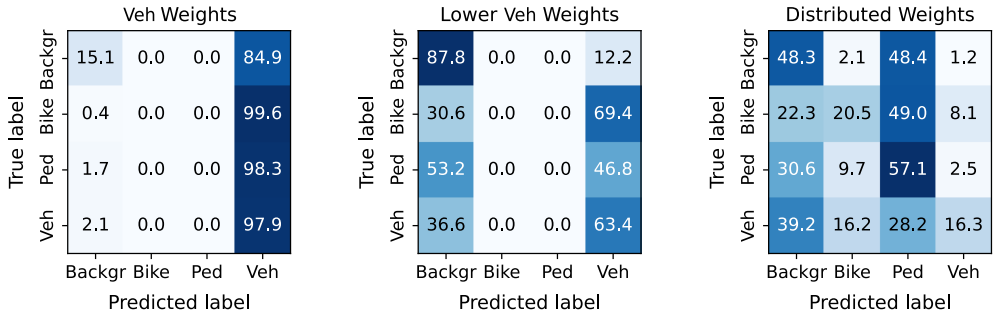


Figure 7. Segmentation confusion matrices for different class weight configurations of the data set classes.

4.2. All Vehicle Data Set

In the following, we therefore limit the segmentation task, distinguishing between vehicles and background in the *All Vehicle Data Set*. The left column of Table 1 shows the resulting F1-score for different model configurations on this data set. The different network architectures are trained with the same hyper parameters to enable comparability. Due to the low data density in the underlying data set, a fitting parameter set needs to be established first to generate reasonable results for any configuration. Due to the high class imbalance, a vanilla network does not produce a reasonable learning result as all classes would just be segmented as background. However, after a fitting configuration is found it becomes evident that the different models reach more or less the same result. While experimenting with different learning strategies, we found that there is no clear ranking between the models, but that the performance levels out in the range of the F1-scores shown in Table 1. It seems that the radar data basis does not include enough discriminative features to separate the classes further. The low data density and quality limits the performance for the proposed models.

Table 1. Segmentation macro-averaged F1-scores on the nuScenes validation data set.

Network	All Vehicle Data Set	Moving Vehicle Data Set
PointNet++	59.91%	75.83%
KPConv	59.88%	74.68%
KPConv-LSTM	59.69%	75.81%
KPConv-CLSTM	60.05%	75.42%
KPLSTM	57.89%	75.34%
KPconv w/o vel	52.36%	52.15%

4.3. Moving Vehicle Data Set

Additionally, we evaluate the models for segmenting moving vehicles against the background in the *Moving Vehicle Data Set*. The moving vehicle class compromises all bounding boxes with the nuScenes attribute *vehicle.moving*. This includes cars, trucks and buses. Table 1 shows the results for this data in the right column.

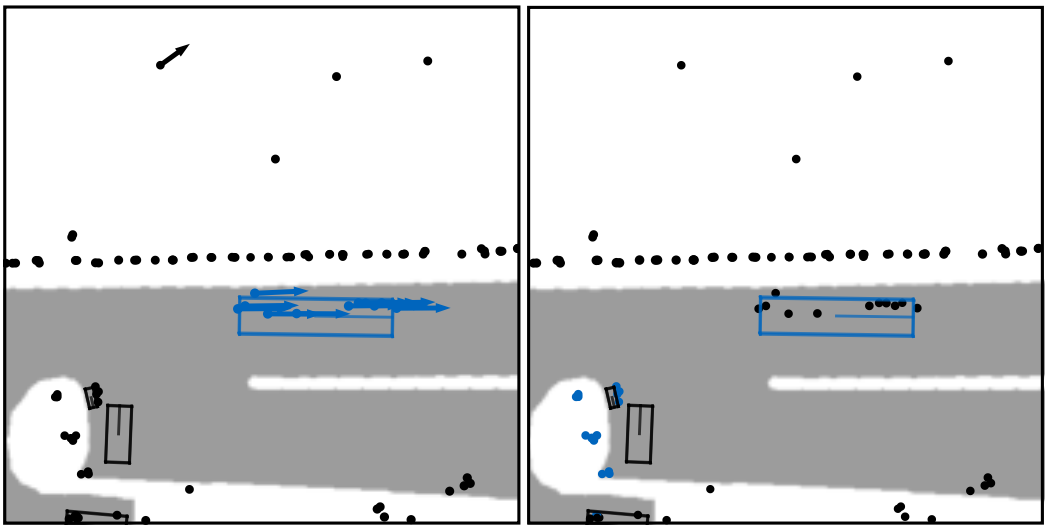
The results show that moving vehicles can be distinguished from the background more successfully than static vehicles, due to the overall higher F1-score achieved. At the same time, the performance levels out at higher overall scores for this type of input data.

4.4. Velocity Input Feature Study

As the velocity component is a strong feature used in classical radar processing, we evaluate its relevance to the result of our segmentation model. We retrain our KPConv model with a reduced input feature space without velocity information. The results are shown in the last row of Table 1. Without the velocity information, the F1-scores are

significantly lower for both data sets. Especially for segmenting moving vehicles, the velocity dimension seems to be a discriminative feature. Without using the strong velocity feature, the segmentation quality of moving and static vehicles reaches a similar score.

Figure 8 shows a qualitative comparison of the segmentation result of a moving bus with and without the velocity feature dimensions. The radar returns from within the bounding box are segmented correctly in Figure 8a. The radar return in the top of the figure, which also comprises a non-zero relative velocity, is correctly segmented as background. Figure 8b shows that the segmentation failed when not using velocity features. Points in the lower left part of the figure are segmented as a moving vehicle. Geometrically, these points show a great similarity to the points of the moving vehicle or even the wall along the road. The RCS feature is not discriminative enough to compensate for this fact.



(**a**) (**b**)

Figure 8. Classification of a moving vehicle. Ground-truth bounding boxes and radar classifications are color-coded. Blue: moving vehicle. Black: rest. The segmentation in (**a**) is more precise than the segmentation in (**b**) due to the inclusion of velocity features. NuScenes sample token: 9a7dc9c8adce4ae68ca26ad5b5f93366.

5. Discussion

A definite conclusion of the model performance cannot be inferred from these results above. The data quality constrains the model performance, meaning that the potential of the KPConv models, as proven for lidar data, could not be shown for the underlying radar data. However, the main reason for this does not seem to lie in the model itself but rather in the data. In recent publications of radar deep-learning processing, the data source is only mentioned briefly. However, many important considerations, when dealing with radar data and especially the particularities of the nuScenes data set, are not discussed in the literature. We believe there is a need to shift the focus from solely the model architectures to the data itself to develop functioning radar deep-learning applications in public research. Therefore, in the following discussion, we not only carefully assess our model, but add a focus on the data side and discuss barriers for the widespread application of machine learning techniques for radar data.

5.1. Model Critique

Due to the low information content in the underlying radar data, model training requires a careful calibration. Although higher segmentation scores could not be achieved, slight variations in learning rate could result in models barely outperforming a guessing model. The low radar density is partly the result of the radar processing itself. In the range-velocity dimension of the radar cube, moving objects are more differentiable from the

static background than static vehicles. Consequently, many labeled static vehicle bounding boxes do not include any or only a few radar points. This is one reason the moving vehicle segmentation performed better than the static case.

The presented LSTM models, in comparison to [21], do not forget old points once an intermediate memory point cloud is full. The data is added or forgotten by the memory point cloud according to the LSTM structure. We assume this to be an advantage of the presented architecture. However, due to the different underlying data source, a direct comparison of the models is not possible here.

We experimented with different functions to associate consecutive radar point clouds in the KPLSTM cell. In general, we would expect learning methods to provide the most accurate data association capability on noisy radar data. However, nearest neighbor sampling outperformed learning-based methods on the presented data set. We assume that the learning of an appropriate association function is not feasible due to the corruption of the radar data in the data set. A detailed study of different learning methods for data association is postponed until an appropriate data is available.

Radar data comes with a class imbalance towards the background class. Most radar reflections originate from non-traffic participants, which makes a high accuracy score possible through the classification of all points as background. The class-weighted focal loss function helps mitigate this problem, by assigning strongly misclassified examples of the minority classes with a higher weight during backpropagation.

At this stage, it is difficult to pinpoint whether the model or the data has the greatest effect on drawbacks of the model performance. To further study the discriminative content in the data set, we overfit the KPConv model to two select scenes from the nuScenes data set. When training a model to segment static and moving vehicles alike, a perfect score is not achieved within 10,000 epochs of training with the same scene. Figure 9a shows the resulting confusion matrix for a *All Vehicle* configuration. Figure 10 shows an extract from this scene. The radar returns of the vehicles have a high similarity to points returned from the background, making even overfitting to the data a challenging task.

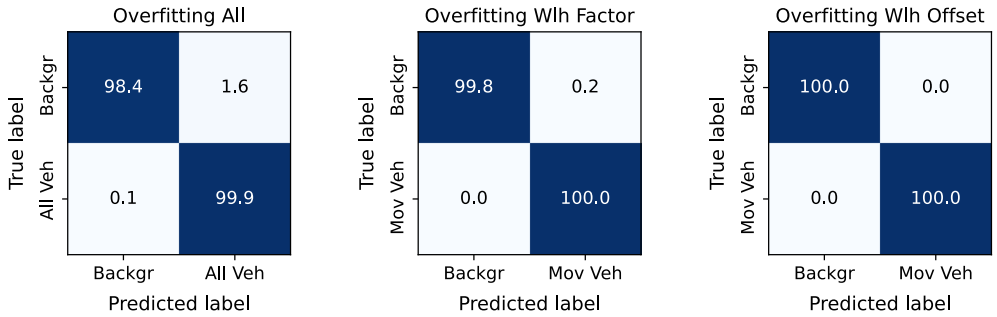


Figure 9. Overfitting confusion matrices. (**a**) Perfect overfitting is not achieved for the *All Vehicle* configuration scene on the left. (**b**) The *Moving Vehicle* data scene is not perfectly segmented when a *wlh*-factor is used. (**c**) Training with the same scene as in (**b**) results in a perfect score when an adequate *wlh*-tolerance is used.

For moving vehicle segmentation, we were able to achieve a perfect segmentation for a scene of a vehicle following scenario. Figure 9b,c shows the confusion matrices for this scene with two different compensations methods for the spatial misalignment. Please note that it was only possible to achieve a perfect score when a *wlh*-tolerance was applied in Figure 9c. When a *wlh*-factor was used in Figure 9b, not all points from the moving object class were correctly associated with the ground-truth bounding box. The model learned to segment these points to the moving vehicle class due to its similarity to points inside the ground-truth box; however, they count negatively towards the metric score.

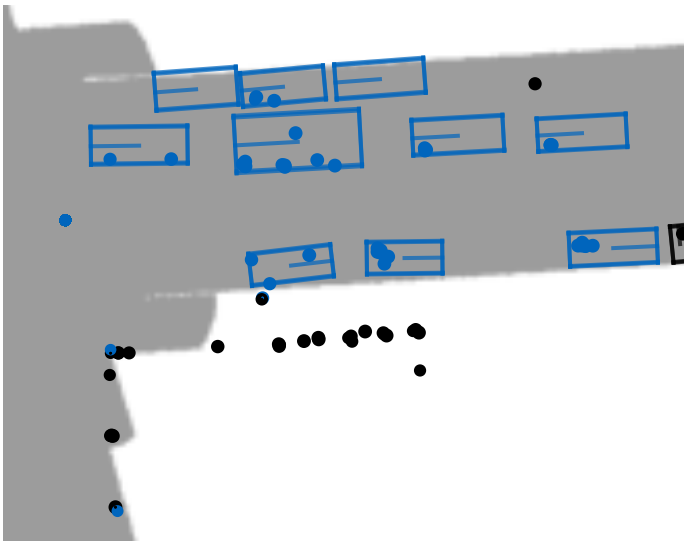


Figure 10. Even in the overfitting scenario, misclassifications are present when static objects are segmented. NuScenes sample token: 205c60f1248343a09bf4c6b6c05d8525.

Labeling bounding boxes from radar data directly is a challenging task itself due to the data sparsity. Nonetheless, if we want to create machine learning models for radar data for a real-world application, it would be beneficial to adapt the lidar bounding boxes to the radar data during the labeling and have sets of ground-truth labels for the sensor modalities independently. Thus far, the *wlh*-tolerance has helped mitigate the problem, but this cannot be the solution when highly precise results are necessary in production.

For the overall performance, further studies with selected data would be necessary to investigate the source of errors in the segmentation result. Due to the overall data quality of the nuScenes radar data, however, it is debatable whether the creation of a high-quality subset of the data set is worth the effort or even achievable. The next section discusses this in detail and motivates our choice to refrain from further optimization due to the quality of the training data.

5.2. Data

This section discusses the particularities of radar data and the nuScenes data set. Data sparsity is a general difficulty when dealing with radar data. Though [21] shows favorable results on a private data set, it becomes clear that even with high-quality-high-resolution input data, the challenge of general semantic segmentation on automotive radar data is far from solved.

In 2019, a review [30] stated that they expect more radar data sets to be released in the future. However, at the start of 2021, only nuScenes [2], Astyx HiRes2019 [31] and DENSE [32] data sets provide point level radar data with 3D bounding box annotations. The HiRes2019 data consists of only 546 frames, which makes training of a learning model on this data impractical. The DENSE data set uses an outdated radar model with even greater sparsity than the nuScenes data set while providing fewer labels. The nuScenes data set consists of 40,000 annotated frames of a production radar. It is the only data set that contains the consecutive samples needed for training time-dependent models. In conclusion, this is the only reasonable public data set for deep-learning applications of radar data.

Despite its unique position, nuScenes has some major drawbacks regarding its radar data. Scheiner [33] compared the performance of their models on high-resolution and production radar data. They state that the nuScenes radar density is far worse than that of their production radar. A comparison of the specifications of the radars of [2,33] and shows that they use a comparable or even the same radar model as nuScenes. The huge differences in data density between the two data sets cannot be explained by the radar

hardware alone. We calculated that over 40% of the annotated vehicles are not covered by any radar detection when considering three radar sweeps for a nuScenes sample. For vehicles labeled as moving, the same holds true for over 24% of the labeled bounding boxes. Consequently, bounding boxes of objects detected by the radar sensor, include fewer point returns than expected from this sensor type. In the following, we look at some factors that decrease the suitability of the nuScenes data set for the development of radar algorithms and conclude with the aspects that a radar data set should consider.

The nuScenes data set includes information about the number of radar points per annotation. However, this number does not distinguish between its five different radar modules. The labels include information about whether or not a vehicle is moving. Although this information is correct for many cases, by manually examining random samples of the data set, we encountered a significant number of annotations where moving objects were labeled as static objects and vice versa. This not only negatively impacts the training, but also impacts the evaluation scores of moving object detection, which is the main application of radar data. Figure 11 shows the ground-truth classes from a scene of a bus moving towards the sensor vehicle. Even though the bus is approaching at a high speed and is close to the sensor vehicle, it is wrongly labeled as a non-moving vehicle.

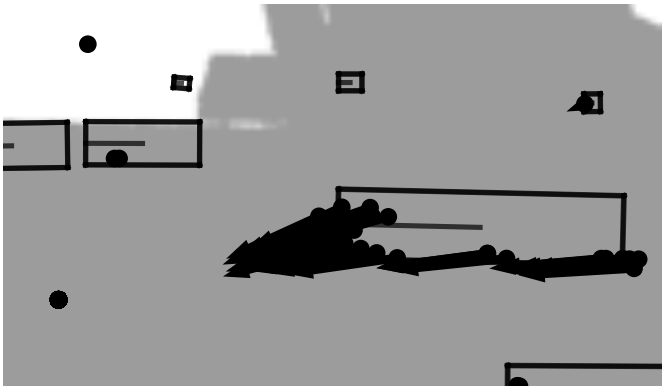


Figure 11. A bus approaching the sensor vehicle in BEV projection. The black color of the ground-truth points and bounding box indicates a static vehicle, despite the bus clearly approaching as indicated by its measured relative velocity. NuScenes sample token: 87a927d7e61345d3a098eb66908bddcb.

The radar data is limited to 125 detections per sweep by the sensor interface. Additional detection filters are applied in the standard configuration of the data set API of nuScenes. The hardware limitation to 125 detections is enforced by cutting off detection signals at an *RCS* value of less than $-5\,\mathrm{dB\,m^2}$. The works of Yamada and Yasugi [34,35] measure the expected *RCS* value of a pedestrian at a distance of less than 10 m to be less than $-8\,\mathrm{dB\,m^2}$ for 76 GHz and 79 GHz radar sensors, respectively. The chosen cut-off value thereby likely filters out many pedestrian detections that cannot be used for training even when the additional filter settings in the data set API are deactivated. By visualizing the data set, we found that not only pedestrian detections are affected by this, but also passenger vehicle are often not detected in the data set. Figure 12 shows a vehicle in line-of-sight direction from the radar sensor. However, no radar points are present inside or near the ground-truth bounding box.

(**a**) (**b**)

Figure 12. Vehicle in line-of-sight direction from the radar sensor. Despite its exposed position, it is not detected with the sensor settings used. (**a**) The vehicle in camera view in front of the sensor vehicle. (**b**) BEV: Zoom onto the GT bounding box in blue and the radar data in the surroundings. No radar return originates from the vehicle. NuScenes sample token: bbba31d91e334f4abf6d1f96bf699980.

While introducing the *wlh*-tolerance, we mentioned that the ground-truth labels are not always accurate regarding the radar returns. Radar reflections seem to originate from outside the ground-truth bounding boxes. The *wlh*-tolerance can mitigate this problem but not completely erase it. No matter the choice of a bounding box tolerance, there will always be returns that lie outside of it and will either "confuse" the learning algorithm or count negatively towards the segmentation score, while seeming to be classified correctly intuitively.

The biggest artificial barrier for successful training that we encountered is the absence of labels for objects that are clearly visible in radar view. Figure 13 shows several moving vehicles from a BEV perspective. Although the radar points on the right are correctly labeled as moving vehicles, the vehicles further to the left in the scene, on both sides of the road, are not labeled. Their presence can be estimated from the point clusters with non-zero velocity. The nuScenes labels are created from lidar view. The resulting number of samples with missing labels for radar data limits the data set quality.

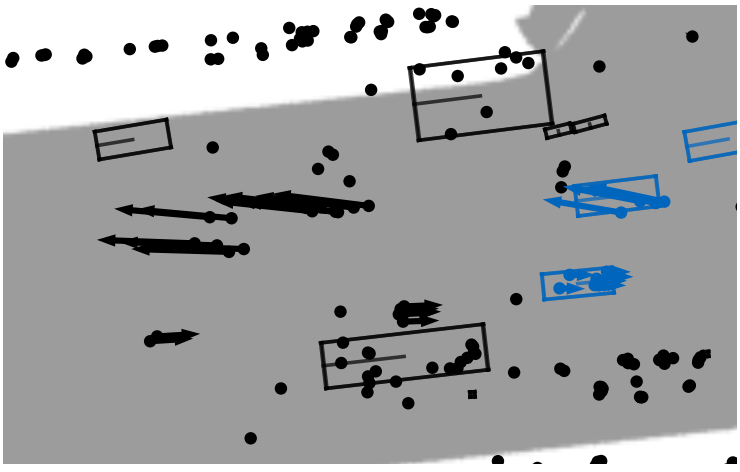


Figure 13. Moving vehicles ground-truth color-coded in blue. Ground-truth bounding boxes are not labeled for four vehicles in the sensor field of view, though they are clearly distinguishable by their relative velocity. NuScenes sample token: e2c7c91b4ea2462090d866539ff6b9e5.

The nuScenes data set is a potent data set for measuring the quality of lidar and camera detection algorithms. The effects mentioned decrease the data quality for the training of radar algorithms significantly. Nonetheless, it is an important contribution to the radar community as it remains the only usable 3D-labeled source of radar point cloud data available to the public. Although it can serve as a comparison benchmark for radar detection algorithms, it cannot measure the absolute performance to be expected

when the algorithms are applied to a real-world use case with a comparable radar module. Slight corrections, such as using the *wlh*-tolerance or filtering for samples with high data density, can be performed to increase the data quality. The creation of a production quality data set, however, would require manual filtering of the instances and possibly even relabeling to create a generally applicable sub data set. This kind of complete revision of the labels seems unreasonable, considering the low raw radar data quality in the data set.

We therefore see a great demand for the release of a data set tailored for radar object detection. This would make it easier to compare different radar processing approaches. The absolute model performance that is measured on private data sets can hardly serve as a metric for comparison due to the huge impact of the underlying data set on the segmentation result. In an ideal data set, the dimensions, the classes and movement properties of the objects are a focus of the scene selection and have been critically revised. A high-resolution radar should be used and its calibration evaluated before data recording. Filtering of radar points, when required by the sensor interface, should be performed, e.g., along the longitudinal and lateral distance dimension to include all nearby objects of interest in the data. As a uniform data density is preferred for model training, sensors around the vehicle should possess little field of view overlap. If sensor overlap is preferred for important directions, e.g., in front of the vehicle, these areas should be separated from remaining areas during training and inference to preserve the homogeneity of the data. lidar data can be used to help the labeling process, though the final labels should be set regarding the radar data.

6. Conclusions and Outlook

This paper develops and benchmarks KPConv-based segmentation networks on radar data. Due to the sparsity of radar data, we motivate the use of direct point processing and one-vs-all segmentation strategies. Furthermore, we model the time dependency during point feature encoding within a KPLSTM cell and in the global feature space with standard LSTM cells. Despite their theoretical motivation, the models could not outperform a pointnet-based network on the underlying data set. Advantages and disadvantages of models are discussed. An in-depth analysis of the data set shows that the public data source itself is a major performance barrier for the models. Absolute performance metrics cannot be measured on currently available public radar data sets. The slower progress of development for radar processing in comparison to lidar data processing can partly be attributed to the lack of publicly available data. Building a high-quality data set of automotive radar data is therefore a major steppingstone in accelerating the progress of radar processing development. We expect the proposed models to show their increased potential when high-quality radar data is available. For that, we make the code for the proposed network architectures and the interface to the nuScenes data set available to the public at: https://github.com/TUMFTM/RadarSeg (accessed on 6 March 2021).

Author Contributions: F.N. as the first author, initiated the idea of this paper and contributed essentially to its concept and content. Conceptualization, F.N. and F.F.; methodology, F.N. and F.F.; software, F.F. and F.N.; data curation, F.F. and F.N.; writing—original draft preparation, F.N.; writing—review and editing, F.F., J.B. and M.L.; visualization, F.N. and F.F.; project administration, J.B. and M.L. All authors have read and agreed to the published version of the manuscript.

Funding: We express gratitude to Continental Engineering Services for funding for the underlying research project.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Geiger, A.; Lenz, P.; Stiller, C.; Urtasun, R. Vision meets Robotics: The KITTI Dataset. *Int. J. Robot. Res.* **2013**, *32*, 1231–1237. [CrossRef]
2. Caesar, H.; Bankiti, V.; Lang, A.H.; Vora, S.; Liong, V.E.; Xu, Q.; Krishnan, A.; Pan, Y.; Baldan, G.; Beijbom, O. nuScenes: A multimodal dataset for autonomous driving. In Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR2020), Washington, DC, USA, 16–18 June 2020.
3. Yoneda, K.; Suganuma, N.; Yanase, R.; Aldibaja, M. Automated driving recognition technologies for adverse weather conditions. *IATSS Res.* **2019**, *43*, 253–262. [CrossRef]
4. Wu, B.; Zhou, X.; Zhao, S.; Yue, X.; Keutzer, K. SqueezeSegV2: Improved Model Structure and Unsupervised Domain Adaptation for Road-Object Segmentation from a lidar Point Cloud. In Proceedings of the International Conference on Robotics and Automation (ICRA), Montreal, QC, Canada, 20–24 May 2019.
5. Milioto, A.; Vizzo, I.; Behley, J.; Stachniss, C. RangeNet ++: Fast and Accurate lidar Semantic Segmentation. In Proceedings of the 2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Macau, China, 3–8 November 2019; IEEE: Piscataway, NJ, USA, 2019; pp. 4213–4220. [CrossRef]
6. Qi, C.R.; Su, H.; Mo, K.; Guibas, L.J. PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, HI, USA, 21–26 July 2017.
7. Qi, C.R.; Yi, L.; Su, H.; Guibas, L.J. PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space. *arXiv* **2017**, arXiv:1706.02413.
8. Thomas, H.; Qi, C.R.; Deschaud, J.E.; Marcotegui, B.; Goulette, F.; Guibas, L.J. KPConv: Flexible and Deformable Convolution for Point Clouds. In Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV), Seoul, Korea, 27 October–2 November 2019.
9. Liang, Z.; Zhang, M.; Zhang, Z.; Zhao, X.; Pu, S. RangeRCNN: Towards Fast and Accurate 3D Object Detection with Range Image Representation. *arXiv* **2020**, arXiv:2009.00206.
10. Shi, S.; Guo, C.; Jiang, L.; Wang, Z.; Shi, J.; Wang, X.; Li, H. PV-RCNN: Point-Voxel Feature Set Abstraction for 3D Object Detection. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Seattle, WA, USA, 14–19 June 2020.
11. Zhou, Y.; Tuzel, O. VoxelNet: End-to-End Learning for Point Cloud Based 3D Object Detection. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–22 June 2018.
12. Yang, Z.; Sun, Y.; Liu, S.; Shen, X.; Jia, J. STD: Sparse-to-Dense 3D Object Detector for Point Cloud. In Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV), Seoul, Korea, 27 October–2 November 2019.
13. Hochreiter, S.; Schmidhuber, J. Long short-term memory. *Neur. Comput.* **1997**, *9*, 1735–1780. [CrossRef] [PubMed]
14. Finn, C.; Goodfellow, I.; Levine, S. Unsupervised Learning for Physical Interaction through Video Prediction. *arXiv* **2016**, arXiv:1605.07157.
15. Lu, Y.; Lu, C.; Tang, C.K. Online Video Object Detection Using Association LSTM. In Proceedings of the 2017 IEEE International Conference on Computer Vision (ICCV), Venice, Italy, 22–29 October 2017; pp. 2363–2371. [CrossRef]
16. Fan, H.; Yang, Y. PointRNN: Point Recurrent Neural Network for Moving Point Cloud Processing. *arXiv* **2019**, arXiv:1910.08287.
17. Schumann, O.; Wohler, C.; Hahn, M.; Dickmann, J. Comparison of Random Forest and Long Short-Term Memory Network Performances in Classification Tasks Using Radar. In Proceedings of the 2017 Symposium on Sensor Data Fusion: Trends, Solutions, Applications (SDF), Bonn, Germany, 10–12 October 2017; IEEE: Piscataway, NJ, USA, 2017; pp. 1–6. [CrossRef]
18. Ester, M.; Kriegel, H.P.; Sander, J.; Xu, X. A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise. In Proceedings of the Second International Conference on Knowledge Discovery and Data Mining, Portland, OR, USA, 2–4 August 1996; pp. 226–231.
19. Lombacher, J.; Hahn, M.; Dickmann, J.; Wohler, C. Potential of radar for static object classification using deep learning methods. In Proceedings of the 2016 IEEE MTT-S International Conference on Microwaves for Intelligent Mobility (ICMIM), San Diego, CA, USA, 19–20 May 2016; pp. 1–4. [CrossRef]
20. Schumann, O.; Hahn, M.; Dickmann, J.; Wohler, C. Semantic Segmentation on Radar Point Clouds. In Proceedings of the 2018 21st International Conference on Information Fusion (FUSION), Cambridge, UK, 10–13 July 2018; IEEE: Piscataway, NJ, USA, 2018; pp. 2179–2186. [CrossRef]
21. Schumann, O.; Lombacher, J.; Hahn, M.; Wohler, C.; Dickmann, J. Scene Understanding With Automotive Radar. *IEEE Trans. Intell. Veh.* **2020**, *5*, 188–203. [CrossRef]
22. Palffy, A.; Dong, J.; Kooij, J.F.P.; Gavrila, D.M. CNN based Road User Detection using the 3D Radar Cube. *IEEE Robot. Automat. Lett.* **2020**, *5*, 1263–1270. [CrossRef]
23. Danzer, A.; Griebel, T.; Bach, M.; Dietmayer, K. 2D Car Detection in Radar Data with PointNets. In Proceedings of the 2019 IEEE Intelligent Transportation Systems Conference (ITSC), Auckland, New Zealand, 27–30 October 2019.
24. Chadwick, S.; Maddern, W.; Newman, P. Distant Vehicle Detection Using Radar and Vision. In Proceedings of the 2019 International Conference on Robotics and Automation (ICRA), Montreal, QC, Canada, 20–24 May 2019.
25. Nobis, F.; Geisslinger, M.; Weber, M.; Betz, J.; Lienkamp, M. A Deep Learning-based Radar and Camera Sensor Fusion Architecture for Object Detection. In Proceedings of the Sensor Data Fusion: Trends, Solutions, Applications (SDF), Bonn, Germany, 15–17 October 2019; pp. 1–7. [CrossRef]

26. Yang, B.; Guo, R.; Liang, M.; Casas, S.; Urtasun, R. RadarNet: Exploiting Radar for Robust Perception of Dynamic Objects. In Proceedings of the European Conference on Computer Vision, Glasgow, UK, 23–28 August 2020.
27. Shi, X.; Chen, Z.; Wang, H.; Yeung, D.Y.; Wong, W.K.; Woo, W.C. Convolutional LSTM Network: A Machine Learning Approach for Precipitation Nowcasting. In Proceedings of the 28th International Conference on Neural Information Processing Systems, Montreal, QC, Canada, 7–12 December 2015.
28. Fent, F. Machine Learning-Based Radar Point Cloud Segmentation. Master's Thesis, Technische Universität München, Munich, Germany, 2020.
29. Lin, T.Y.; Goyal, P.; Girshick, R.; He, K.; Dollár, P. Focal Loss for Dense Object Detection. In Proceedings of the IEEE International Conference on Computer Vision (ICCV), Venice, Italy, 22–29 October 2017.
30. Feng, D.; Haase-Schuetz, C.; Rosenbaum, L.; Hertlein, H.; Duffhauss, F.; Glaser, C.; Wiesbeck, W.; Dietmayer, K. Deep Multi-modal Object Detection and Semantic Segmentation for Autonomous Driving: Datasets, Methods, and Challenges. *IEEE Trans. Intell. Transp. Syst.* **2021**, *22*, 1341–1360. [CrossRef]
31. Meyer, M.; Kuschk, G. Automotive Radar Dataset for Deep Learning Based 3D Object Detection. In Proceedings of the 2019 16th European Radar Conference (EuRAD), Paris, France, 2–4 October 2019; pp. 129–132.
32. Bijelic, M.; Gruber, T.; Mannan, F.; Kraus, F.; Ritter, W.; Dietmayer, K.; Heide, F. Seeing Through Fog Without Seeing Fog: Deep Multimodal Sensor Fusion in Unseen Adverse Weather. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Seattle, WA, USA, 13–19 June 2020.
33. Scheiner, N.; Schumann, O.; Kraus, F.; Appenrodt, N.; Dickmann, J.; Sick, B. Off-the-shelf sensor vs. experimental radar—How much resolution is necessary in automotive radar classification? In Proceedings of the 2020 IEEE 23rd International Conference on Information Fusion (FUSION), Rustenburg, South Africa, 6–9 July 2020.
34. Yamada, N.; Tanaka, Y.; Nishikawa, K. Radar cross section for pedestrian in 76GHz band. In Proceedings of the Microwave Conference, Paris, France, 4–6 October 2005; IEEE: Piscataway, NJ, USA, 2005; pp. 4–1018. [CrossRef]
35. Yasugi, M.; Cao, Y.; Kobayashi, K.; Morita, T.; Kishigami, T.; Nakagawa, Y. 79GHz-band radar cross section measurement for pedestrian detection. In Proceedings of the Asia-Pacific Microwave Conference proceedings (APMC), Seoul, Korea, 5–8 November 2013; IEEE: Piscataway, NJ, USA, 2013; pp. 576–578. [CrossRef]