2023 IEEE International Conference on Robotics and Automation (ICRA 2023)
May 29 - June 2, 2023. London, UK

NVRadarNet: Real-Time Radar Obstacle and Free Space Detection for Autonomous Driving

Alexander Popov, Patrik Gebhardt, Ke Chen, Ryan Oldja,
Heeseok Lee, Shane Murray, Ruchi Bhargava, Nikolai Smolyanskiy
NVIDIA

Abstract— **Detecting obstacles is crucial for safe and efficient autonomous driving. To this end, we present NVRadarNet, a deep neural network (DNN) that detects dynamic obstacles and drivable free space using automotive RADAR sensors. The network utilizes temporally accumulated data from multiple RADAR sensors to detect dynamic obstacles and compute their orientation in a top-down bird's-eye view (BEV). The network also regresses drivable free space to detect unclassified obstacles. Our DNN is the first of its kind to utilize sparse RADAR signals in order to perform obstacle and free space detection in real time from RADAR data only. The network has been successfully used for perception on our autonomous vehicles in real self-driving scenarios. The network runs faster than real time on an embedded GPU and shows good generalization across geographic regions.** [1]

I. INTRODUCTION

The ability to detect dynamic and stationary obstacles (e.g., cars, trucks, pedestrians, bicycles, hazards) is critical for autonomous vehicles. This is particularly important in semi-urban and urban settings characterized by complex scenes with large amounts of occlusion and varieties of shapes.

Previous perception methods rely heavily on utilizing cameras [1] [2] [3] or LiDARs [4] [5] [6] [7] to detect obstacles. These methods have a number of drawbacks: they are unreliable in cases of heavy occlusions, the sensors may be prohibitively expensive, they can be unreliable in adverse weather conditions [8] or at night. Traditional RADAR based obstacle detection methods work well in detecting moving objects that have good reflection properties, but tend to struggle when estimating object dimensions and orientations and can often fail in detecting stationary objects or objects with poor RADAR reflectivity.

In this paper, we present a deep neural network (DNN) that detects moving and stationary obstacles, computes their orientation and size, and detects drivable free space from RADAR data alone. We do this in top-down bird's-eye view (BEV) for highway and urban scenarios while using readily available automotive RADARs. Our method relies on RADAR peak detections alone [9] [10] since automotive RADAR firmware provides only this data. In contrast, other approaches [11] [12], require expensive Fast Fourier Transformation operations on the raw RADAR data cube cross-sections, which are often not available in commercial automotive sensors.

[1] Video at `https://youtu.be/WlwJJMltoJY` .

Our deep learning approach is able to accurately distinguish between stationary obstacles, such as cars, and stationary background noise. This is important when navigating in a cluttered urban environment. In addition, our approach allows us to regress the dimensions and orientations of these obstacles, which classical methods cannot provide. Our DNN can even detect obstacles with poor reflectivity like pedestrians. Finally, our method provides an occupancy probability map to mark unclassified obstacles and regresses drivable free space.

We have tested our NVRadarNet DNN in real-world autonomous driving on our vehicles running NVIDIA DRIVE AGX's embedded GPU. Our DNN runs *faster than real-time* at **1.5 ms** end-to-end and provides sufficient time for the planner to react safely.

Our contributions are as follows:

- NVRadarNet: A first of its kind multi-class deep neural network that detects dynamic and stationary obstacles end-to-end without post-processing in a top-down bird's-eye view (BEV) using only peak detections coming from automotive RADARs;
- A novel semi-supervised drivable free space detection method using only RADAR peak detections;
- A DNN architecture that runs *faster than real-time* at **1.5 ms** end-to-end on an embedded GPU.

II. PREVIOUS WORK

Obstacle Detection. Fast and efficient obstacle perception is a core component of a self-driving vehicle. Automotive RADAR sensors provide a cost-efficient way of obtaining rich 3D positioning and velocity information and are widely available on most modern cars. Several recent papers examined the use of the dense RADAR data cubes in order to perform obstacle detection [11] [12]. However, these methods require high input/output bandwidth to obtain such rich data, making them difficult to implement in real-world autonomous vehicles. Thus, most classical methods in automotive RADAR applications utilize post-processed peak detections from the data cube in order to perform classification and occupancy grid detection [13] [14] [15]. Others realized that the RADAR peak detections can be viewed as a sparse 3D point cloud and therefore can be used in sensor fusion along with 3D LiDAR points in approaches similar to LiDAR DNNs [4] [16] [17] [5] [18]. There were

979-8-3503-2365-8/23/$31.00 ©2023 IEEE

also attempts to enhance camera 3D obstacle detection by fusing it with RADAR [19].

Free Space Detection. RADAR-based drivable free space estimation has been attempted in [20] and [21].

Our DNN performs multi-class detection of dynamic and static obstacles together with the segmentation of drivable free space by using RADAR peak detections alone. Our DNN architecture is lightweight and runs *faster than real-time* at **1.5 ms** end-to-end on an embedded GPU (on NVIDIA DRIVE AGX). It has been proven to be robust in real-world driving and was tested on over 10000 km of highway and urban roads as part of our autonomous stack. To date, we are not aware of any RADAR peak detections only DNN that can perform all of these tasks and can run efficiently on autonomous vehicles.

III. METHOD

A. Input Generation

The input to our network is a top-down BEV orthographic projection of accumulated RADAR detection peaks around our ego-vehicle, which is placed at the center of this top-down bird's-eye view (BEV) with its front facing right.

To compute this input, we first accumulate RADAR peak detections across all RADAR sensors on our vehicle (8 radars covering 360 degrees field of view, as found in our test fleet) and then transform them to our ego-vehicle rig coordinate system. We also accumulate these peak detections temporally over 0.5 seconds in order to increase the density of the signal. Each data point gets a relative timestamp to indicate its age, similar to [16]. Next, we perform ego-motion compensation for the accumulated detections to the latest known vehicle position. We propagate the older points using the known ego-motion of our vehicle to estimate where they will be at the time of the DNN inference (current time).

Next, we project each accumulated detection to a top-down BEV grid using the desired space quantization to create an input tensor for our DNN. We set our input resolution to 800×800 pixels with $\pm$ 100 m range in each direction, resulting in 25 cm per pixel resolution. Each valid BEV pixel (with data) gets a set of features in its depth channel computed by averaging the raw signal features of the RADAR detections that land in that pixel. Our final input for time t is a tensor $I_t \in \mathbb{R}^{h\times w\times 5}$ where $h = 800, w = 800$ are height and width of a top-down view. The 5 RADAR features in the depth channel are the averages of: Doppler, elevation angle, RADAR cross section (*RCS*), azimuth angle and the relative detection timestamp. We normalize these values to a $[0, 1]$ range for training stability using maximum and minimum values provided by the hardware specifications. The resulting tensor is used as input to our network.

B. Label Propagation

We use LiDAR-based human-annotated bounding box labels as the ground truth for training our RADAR DNN. These labels are created using the LiDAR data for the same scenes on which we train our RADAR DNN. Given how sparse the RADAR signal is, it is practically impossible for humans to distinguish vehicles using RADAR points alone even in the top-down BEV view. Hence, we rely on LiDAR to label training data. We capture both LiDAR and RADAR data at different frequencies and select the data closest in time for processing. We then create a top-down BEV projection of the LiDAR scene for humans to annotate objects with the bounding box labels and free space with polylines. For each labeled LiDAR BEV frame, we compute the closest accumulated RADAR BEV image via the pre-processing method described above and then transfer the labels to the RADAR top-down view. We further clean up the the ground truth by removing any vehicle labels that contain fewer than 4 RADAR detections within a 70 m range. Finally, we remove any detections with an *RCS* below -40 dBm as we found that they introduce more noise than signal. An example can be seen in Fig. 1.

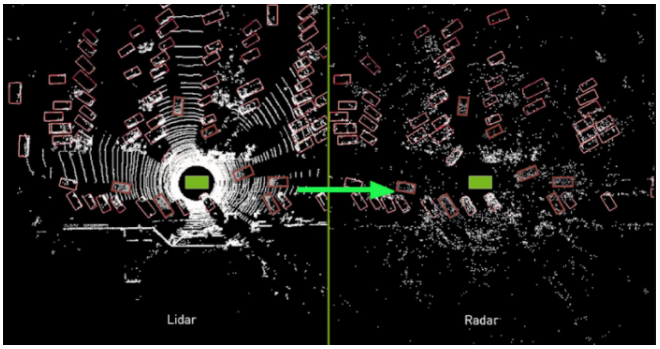

Fig. 1: Propagating bounding box labels for cars from the LiDAR domain to the RADAR domain.

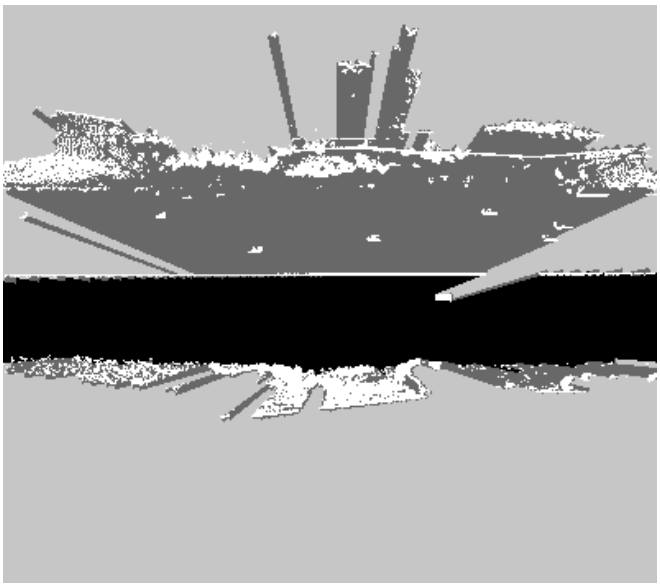
Fig. 2: Visual representation of the free space target: observed and free in black, observed and occupied in white, unobserved in light gray and partially observed in dark gray.

C. Free Space Label Generation

The free space target is generated by using the raw LiDAR point cloud. First, we pre-process the point cloud by identifying and removing the points belonging to the drivable surface itself by using surface slope angle estimation

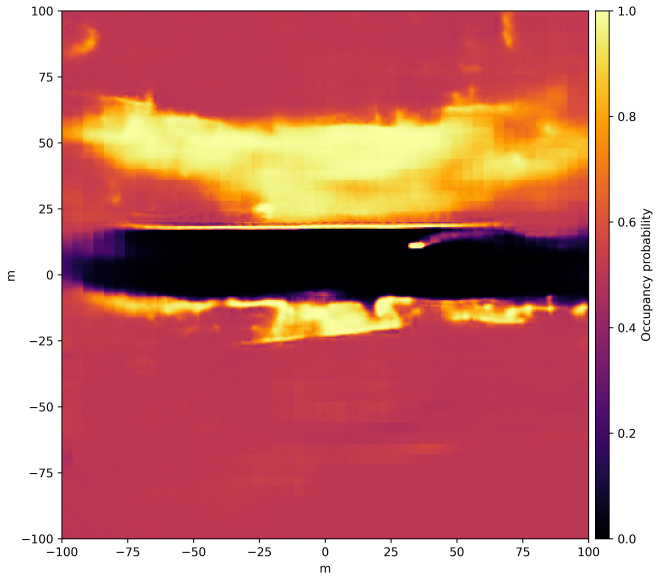

Fig. 3: Inferred dense occupancy probability map, in Cartesian coordinate space, showing probabilities from low in dark to high probability in gradients from red to yellow (highest).

of adjacent LiDAR scan lines. We then overlay manually obtained LiDAR free space labels to further clean up this estimate. Next, a set of rays is traced from the ego-vehicle's origin in all angular directions, enabling us to reason about which regions are:

- Observed and free
- Observed and occupied
- Unobserved
- Partially observed

Finally, we overlay our existing 3D obstacle labels on top of the automatically derived occupancy. We explicitly mark obstacles as observed and occupied. See Fig. 2.

D. Dataset

Our model is trained on a diverse internal dataset with over 300k training frames and over 70k validation frames sampled from hundreds of hours of driving in several geographic regions. The dataset includes a combination of urban and highway data and contains synchronized LiDAR, RADAR and IMU readings. The labels are human annotated and include vehicles, cyclists, pedestrians and drivable free space.

E. Network Architecture

We use a DNN architecture similar to a Feature Pyramid Network [22]. Our DNN consists of encoder and decoder components and several heads for predicting different outputs. See Fig. 4 for high-level structure and Table I for details.

Our encoder starts with a *2D convolutional layer* with 64 filters, stride 2 and 7×7 kernels. It is followed by 4 blocks of 4 layers each, where each block increases the number of filters by two, while dividing the resolution in half. Each layer in the block contains a *2D convolution* with *batch normalization* and *ReLU activation*.

TABLE I: Network architecture for NVRadarNet

Layer	Layer description	Input	Output dimensions
	Inputs:		
input	*Input RADAR data*	–	$5 \times 800 \times 800$
	Encoder:		
1	conv (7×7), $ReLU$	input	$64 \times 400 \times 400$
2a	conv (3×3), $ReLU$	1	$64 \times 200 \times 200$
2b	conv (3×3), $ReLU$	2a	$64 \times 200 \times 200$
3a	conv (3×3), $ReLU$	2b	$64 \times 200 \times 200$
3b	conv (3×3), $ReLU$	3a	$64 \times 200 \times 200$
4a	conv (3×3), $ReLU$	3b	$128 \times 100 \times 100$
4b	conv (3×3), $ReLU$	4a	$128 \times 100 \times 100$
4c	conv (3×3), $ReLU$	4b	$128 \times 100 \times 100$
4d	conv (3×3), $ReLU$	4c	$128 \times 100 \times 100$
5a	conv (3×3), $ReLU$	4d	$256 \times 50 \times 50$
5b	conv (3×3), $ReLU$	5a	$256 \times 50 \times 50$
5c	conv (3×3), $ReLU$	5b	$256 \times 50 \times 50$
5d	conv (3×3), $ReLU$	5c	$256 \times 50 \times 50$
6a	conv (3×3), $ReLU$	5d	$512 \times 50 \times 50$
6b	conv (3×3), $ReLU$	6a	$512 \times 50 \times 50$
6c	conv (3×3), $ReLU$	6b	$512 \times 50 \times 50$
6d	conv (3×3), $ReLU$	6c	$512 \times 50 \times 50$
	Decoder:		
freespace_output	deconv (4×4), $ReLU$	6d	$2 \times 400 \times 400$
regression_output	deconv (4×4), $ReLU$	6d	$6 \times 200 \times 200$
class_output	deconv (4×4), $ReLU$	6d	$4 \times 200 \times 200$

The decoder consists of one *transposed 2D convolution* with stride 4 and 4×4 kernels per head. We also experimented with using two *transposed 2D convolutions* with a skip connection in the middle. The resulting output tensor is at $1/4$ of the spatial resolution of the input.

We use the following heads in our network:

- **Class segmentation head** predicts a multi-channel tensor, one channel per class. Each value contains a confidence indicating that a given pixel belongs to a class corresponding to its channel.
- **Instance regression head** predicts oriented bounding boxes for an object using an n_r ($n_r = 6$) channels of information for each predicted pixel. The n_r element vectors contains: $[\delta_x, \delta_y, w_0, l_0, \sin\theta, \cos\theta]$, where (δ_x, δ_y) points toward the centroid of the corresponding object, $w_0 \times l_0$ are the object dimensions, and θ is the orientation in the top-down BEV.
- **Freespace head** computes a map of occupancy probabilities for each grid cell [20].

F. Loss

Our loss consists of a standard cross-entropy loss for the classification head, with a larger weight emphasis on the minority classes, L_1 loss for instance bounding box regression, and the inverse sensor model loss for free space detection [20].

We combine these losses using Bayesian learned weights by modeling each task weight as a homoscedastic task-dependent uncertainty following the method described in [23]. This approach allows us to efficiently co-train these three diverse tasks without affecting the overall model accuracy.

The total loss is defined as:

$$TotalLoss = \sum_{i=0}^{K-1} L_i w_i + \mu_w \tag{1}$$

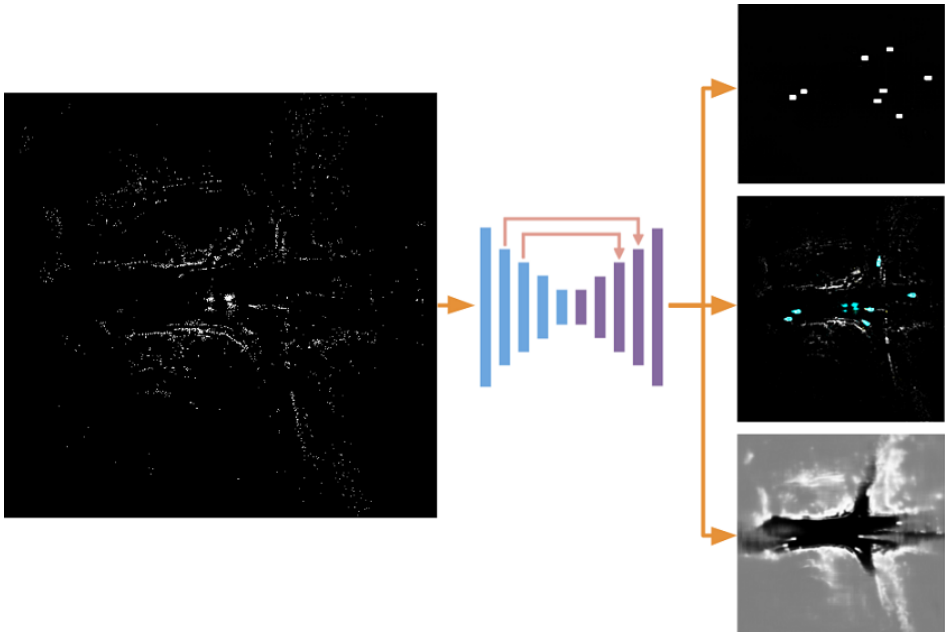

Fig. 4: Network architecture. Our network uses CNN for the encoder and decoder with skip connections. The network has three heads: a classification head (produces detection probabilities), a shape regression head (produces bounding box parameters) and a free space segmentation head.

where K is the number of tasks/heads, L_i is a loss for task i, $w_i = e^{-\delta_i}$, δ_i is a learned log variance parameter per task, and μ_w is the mean of w_i weights.

G. End-to-end Obstacle Detection

In order to avoid expensive non-maximum suppression (NMS) or clustering at post-processing (e.g. DBSCAN), we employ an end-to-end approach by classifying a single pixel per obstacle, as inspired by OneNet [24].

First we compute the L_1 loss for the regression head and the pixel-wise classification loss for the classification head. Next, for each target obstacle, we select the foreground pixel with the lowest total loss between $(\textit{ClassWeight} * \textit{ClassLossPerPixel}) + \textit{RegressionLossPerPixel}$. This pixel is then selected for the final loss computation while the rest of the foreground pixels are ignored. The losses from the background pixels are then selectively used by utilizing hard negative mining. Finally, we perform batch normalization by dividing the total cross-entropy loss by the number of positive pixels selected during the above process. The regression losses are computed only for the selected positive pixels.

At inference time we pick all of the candidate pixels above a certain threshold in the classification head, per class. The obstacle dimensions are picked directly from the regression head for each corresponding threshold candidate.

By using this technique our network is able to directly output the final obstacle without expensive post-processing.

H. Converting ISM Head Output to a Radial Distance Map

Autonomous vehicle applications often represent drivable free space area by its boundary contour. In this sections we describe how to convert the boundary contour to a *radial distance map* (RDM) if needed. The RDM assigns a set of angular directions ϕ_f, in the top-down BEV view around the ego-vehicle, to the distance d_f between a reference point $\vec{p}_{ref}$ on the ego-vehicle and the drivable free space boundary. To compute the RDM, we first re-sample the dense occupancy probability map (DNN output) into a polar coordinate system centered around the reference point $\vec{p}_{ref}$. By employing a nearest-neighbour interpolation schema, the re-sampling process can be expressed in terms of an indexing operation. This assigns the value of each pixel (ϕ_f, d_f) of the polar representation the value of a single pixel of the predicted dense occupancy probability map. Since this mapping only depends on the dimensions of the occupancy map and the position of reference point $\vec{p}_{ref}$, all required indices can be calculated offline and stored in a lookup table. Fig. 5 shows the occupancy probability map. Fig. 3 shows it re-sampled in polar coordinates. After re-sampling, the distance d_f for each angular direction ϕ_f is determined by finding the first pixel along each angular axis, where the occupancy probability reaches some threshold p_{occ}. Fig. 6 shows the RDM representation of the drivable free space boundary derived by this procedure from the dense occupancy probability map shown in Fig. 3.

IV. EXPERIMENTS

To evaluate our method we trained and tested our model on both the open nuScenes [25] and on our internally collected datasets.

A. Internal Dataset Experiments

Datasets, benchmarks and published DNNs dedicated to RADAR based obstacle and freespace detection are limited at this time which presents difficulty when evaluating. See

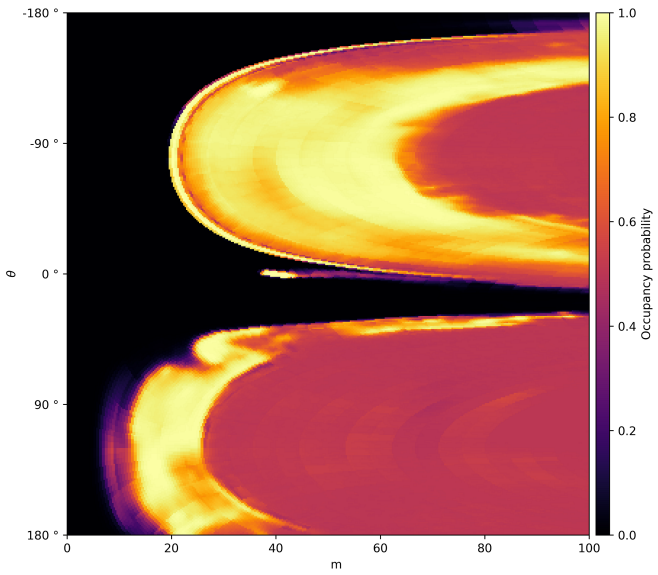

Fig. 5: Predicted dense occupancy map re-sampled into the polar coordinate system centered around the reference point $\vec{p}_{ref}$. Gradient colors from red (low) to yellow (high) show probabilities.

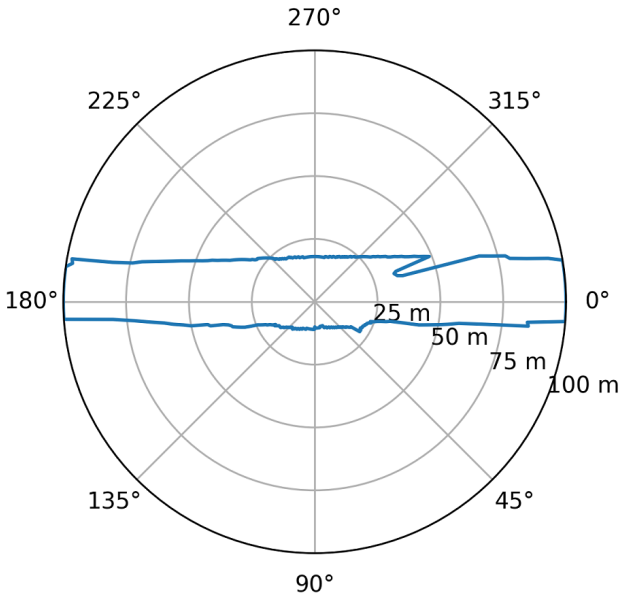

Fig. 6: Radial distance map representation of the drivable free space boundary extracted from the predicted dense occupancy probability map in Cartesian coordinate space.

Table II for a list of available methods and their features. The closest works [17] [5] [19] [18] use sensor fusion and do not share RADAR only results publicly. Thus, to the best of our knowledge we are setting a baseline for obstacle detection, classification and freespace regression using RADAR peaks alone. Detection of pedestrians and cyclists is a big challenge due to the sparsity of the RADAR signal.

We evaluated our DNN on our internal NVIDIA's RADAR dataset as well as on the nuScenes public dataset. We also compared our DNN to other published works as much as was practically possible and list all results in this section.

For our internal NVIDIA's RADAR dataset, we used held

TABLE II: Related RADAR detection methods. Our method (in bold) uses only RADAR data, supports object and free space detection and we provide public results.

Method	**Radar Only**	**Public**	**Objects**	**Freespace**
RadarNet [17]	✗	✓	✓	✗
FishingNet [5]	✗	✓	✓	✗
LiRaNet [18]	✗	✓	✓	✗
RADModel [11]	✓	✗	✓	✗
XSense [12]	✓	✗	✓	✗
OccupancyNet [21]	✓	✓	✗	✓
Ours	✓	✓	✓	✓

TABLE III: Our DNN's obstacle detection accuracy on the internal NVIDIA dataset by class and range.

Class	**Range**	**F-score, small res.**[1]	**F-score, large res.**[2]
vehicles	0 – 10 m	0.728	0.770
vehicles	10 – 25 m	0.608	0.628
vehicles	25 – 40 m	0.728	0.485
vehicles	40 – 70 m	0.327	0.319
vehicles	70 – 100 m	0.225	0.216
pedestrians	0 – 10 m	0.197	0.248
pedestrians	10 – 25 m	0.204	0.24
pedestrians	25 – 40 m	0.145	0.174
pedestrians	40 – 70 m	0.084	0.113
pedestrians	70 – 100 m	0.040	0.062
cyclists	0 – 10 m	0.145	0.264
cyclists	10 – 25 m	0.125	0.257
cyclists	25 – 40 m	0.085	0.192
cyclists	40 – 70 m	0.064	0.137
cyclists	70 – 100 m	0.065	0.114

[1] Input resolution: $800 \times 800 \times 5$.
[2] Input resolution: $1024 \times 1024 \times 5$.

out test data mentioned in section III-D for evaluation. It is important to note that, even after filtering out the ground truth bounding boxes, which contain too few RADAR peak detections (as described in section III-B), we still end up with noisy ground truth labels. For example, there are many instances where vehicles are occluded by other vehicles and so human labelers are not able to create good ground truth from LiDAR data alone. In such cases, RADAR still produces valid returns and some obstacles are correctly classified by our DNN, but are marked as false positives at evaluation. This lowers our precision. Also, LiDAR sensor is mounted higher on the vehicle (roof) compared to the RADAR sensors (bumpers) and therefore some obstacles may be visible by LiDAR but have limited RADAR visibility which leads to noisy RADAR data and labels. This results in false negatives and lowers our recall.

Our results for the object detection task can be seen in Tables III, IV. The metrics for the free space detection task (Table V) are calculated for the free space region and the free space RDM independently. The free space region is defined by an occupancy probability $p_o < 0.4$.

B. NuScenes Dataset Performance

We further train and evaluate our approach on the public nuScenes dataset [25]. This dataset contains sensor data from 1 LiDAR and 5 RADARs. However, the sensors in this dataset are from the older generation making direct comparison difficult. The LiDAR sensor used for the nuScenes

TABLE IV: Our DNN's obstacle detection accuracy on the internal NVIDIA dataset by class.

Class	mAP, small resolution[1]	mAP, large resolution[2]
vehicles	0.438	0.473
pedestrians	0.039	0.057
cyclists	0.032	0.066

[1] Input resolution: $800 \times 800 \times 5$.
[2] Input resolution: $1024 \times 1024 \times 5$.

TABLE V: Our DNN's free space regression accuracy on the internal NVIDIA dataset.

Resolution	Accuracy	IoU	RDM MAE	RDM IoU
$800 \times 800 \times 5$	0.970	0.597	3.129 m	0.630
$1024 \times 1024 \times 5$	0.968	0.576	2.674 m	0.712

data collection contains only 32 beams vs 128 beams in our internal dataset. The extra sparsity in this dataset degrades the quality of our auto-generated free space targets. Similarly, the Continental ARS 408-21 RADARs used in nuScenes dataset produce significantly sparser detections when compared to the newer generation Continental ARS430 RADAR sensors we use in our internal dataset. Nonetheless, we demonstrate respectable results, especially at close ranges. See Tables VI, VII and VIII for details.

We further compare our NVRadarNet DNN free space detection accuracy against the method published in [21], which also presents results on the nuScenes dataset. However, this method operates on a grid covering the area in front of the ego vehicle up to 86 m with 10 m to each side, and unlike our approach it does not regress or classify obstacles. For this comparison we performed the evaluation on the same image region, while converting the predicted occupancy probability to three classes *Occupied*, *Free* and *Unobserved* as follows.

- Occupied: $p_{occ} > 0.65$
- Free: $p_{occ} < 0.35$
- Unobserved: $0.35 <= p_{occ} <= 0.65$

The results are given in Table IX. Our DNN outperforms the other method for occupied space regression (best results in bold) and performs similarly on other tasks.

TABLE VI: Our DNN's obstacle detection accuracy on nuScenes dataset by class and range.

Class	Range	F-score, small[1]	F-score, large[2]
vehicles	0 – 10 m	0.520	0.563
vehicles	10 – 25 m	0.500	0.538
vehicles	25 – 40 m	0.352	0.386
vehicles	40 – 70 m	0.180	0.199
vehicles	70 – 100 m	0.080	0.086
pedestrians	0 – 10 m	0.056	0.059
pedestrians	10 – 25 m	0.046	0.052
pedestrians	25 – 40 m	0.030	0.040
pedestrians	40 – 70 m	0.016	0.024
pedestrians	70 – 100 m	0.000	0.005
cyclists	0 – 10 m	0.050	0.030
cyclists	10 – 25 m	0.068	0.066
cyclists	25 – 40 m	0.059	0.066
cyclists	40 – 70 m	0.044	0.041
cyclists	70 – 100 m	0.000	0.000

[1] Input resolution: $800 \times 800 \times 5$.
[2] Input resolution: $1024 \times 1024 \times 5$.

TABLE VII: Our DNN's obstacle detection accuracy on nuScenes dataset by class.

Class	mAP, small resolution[1]	mAP, large resolution[2]
vehicles	0.245	0.280
pedestrians	0.002	0.003
cyclists	0.005	0.004

[1] Input resolution: $800 \times 800 \times 5$.
[2] Input resolution: $1024 \times 1024 \times 5$.

TABLE VIII: Our DNN's free space regression accuracy on nuScenes dataset.

Resolution	Accuracy	IoU	RDM MAE	RDM IoU
$800 \times 800 \times 5$	0.896	0.351	10.621 m	0.394
$1024 \times 1024 \times 5$	0.881	0.353	9.584 m	0.441

C. NVRadarNet DNN Inference

Our NVRadarNet DNN can be trained in mixed precision mode using INT8 quantization without any loss of accuracy. We export the network using NVIDIA TensorRT and time it on NVIDIA DRIVE AGX's embedded GPU used in our autonomous vehicles. Our DNN is able to achieve **1.5 ms** end-to-end inference with all three heads. We process all of the surround RADARs, perform obstacle detection and free space segmentation much faster than real-time on the embedded GPU. It was difficult to find other RADAR DNNs inference timings in the literature for direct comparison. We only found that [11] is an order of magnitude slower.

V. CONCLUSION

In this work, we presented NVRadarNet DNN, a real-time deep neural network for obstacle and drivable free space detection from raw RADAR data provided by common automotive RADARs. We benchmarked our DNN on both internal NVIDIA dataset and the public nuScenes dataset and provided accuracy results. Our DNN runs faster than real-time at **1.5 ms** end-to-end inference time on NVIDIA DRIVE AGX's embedded GPU. To date, we are not aware of any other RADAR only networks that can simultaneously perform obstacle detection and free space regression while running faster than real-time on automotive embedded computers.

ACKNOWLEDGMENT

We would like to thank Sriya Sarathy, Tilman Wekel, Stan Birchfield for their contributions. We would like to acknowledge David Nister, Sangmin Oh, Minwoo Park for their support.

TABLE IX: Comparison to OccupancyNet [21] on nuScenes dataset. Best results in bold.

Method	Occupied	Free	Unobs.	mIoU
OccupancyNet [21]	0.108	**0.614**	**0.593**	**0.439**
Ours @ 800×800	**0.237**	0.564	0.436	0.412
Ours @ 1024×1024	0.222	0.574	0.405	0.400

REFERENCES

[1] J. Philion and S. Fidler, "Lift, splat, shoot: Encoding images from arbitrary camera rigs by implicitly unprojecting to 3d," in *European Conference on Computer Vision*. Springer, 2020, pp. 194–210.

[2] X. Zhou, D. Wang, and P. Krähenbühl, "Objects as points," *arXiv preprint arXiv:1904.07850*, 2019.

[3] F. Manhardt, W. Kehl, and A. Gaidon, "Roi-10d: Monocular lifting of 2d detection to 6d pose and metric shape," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 2069–2078.

[4] K. Chen, R. Oldja, N. Smolyanskiy, S. Birchfield, A. Popov, D. Wehr, I. Eden, and J. Pehserl, "Mvlidarnet: Real-time multi-class scene understanding for autonomous driving using multiple views," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2020, pp. 2288–2294.

[5] N. Hendy, C. Sloan, F. Tian, P. Duan, N. Charchut, Y. Xie, C. Wang, and J. Philbin, "Fishing net: Future inference of semantic heatmaps in grids," *arXiv preprint arXiv:2006.09917*, 2020.

[6] W. Luo, B. Yang, and R. Urtasun, "Fast and furious: Real time end-to-end 3d detection, tracking and motion forecasting with a single convolutional net," in *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, 2018, pp. 3569–3577.

[7] B. Yang, W. Luo, and R. Urtasun, "Pixor: Real-time 3d object detection from point clouds," in *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, 2018, pp. 7652–7660.

[8] M. Bijelic, T. Gruber, and W. Ritter, "A benchmark for lidar sensors in fog: Is detection breaking down?" in *2018 IEEE Intelligent Vehicles Symposium (IV)*, 2018, pp. 760–767.

[9] R. Wan, Y. Song, T. Mu, and Z. Wang, "Moving target detection using the 2d-fft algorithm for automotive fmcw radars," in *2019 International Conference on Communications, Information System and Computer Engineering (CISCE)*, 2019, pp. 239–243.

[10] V. Winkler, "Range doppler detection for automotive fmcw radars," in *2007 European Microwave Conference*, 2007, pp. 1445–1448.

[11] B. Major, D. Fontijne, A. Ansari, R. T. Sukhavasi, R. Gowaikar, M. Hamilton, S. Lee, S. Grzechnik, and S. Subramanian, "Vehicle detection with automotive radar using deep learning on range-azimuth-doppler tensors," in *2019 IEEE/CVF International Conference on Computer Vision Workshop (ICCVW)*, 2019, pp. 924–932.

[12] X. Dong, P. Wang, P. Zhang, and L. Liu, "Probabilistic oriented object detection in automotive radar," in *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2020, pp. 458–467.

[13] J. Dickmann, N. Appenrodt, J. Klappstein, H.-L. Bloecher, M. Muntzinger, A. Sailer, M. Hahn, and C. Brenk, "Making bertha see even more: Radar contribution," *IEEE Access*, vol. 3, pp. 1233–1247, 2015.

[14] J. Lombacher, M. Hahn, J. Dickmann, and C. Wöhler, "Object classification in radar using ensemble methods," in *2017 IEEE MTT-S International Conference on Microwaves for Intelligent Mobility (ICMIM)*, 2017, pp. 87–90.

[15] R. Prophet, A. Deligiannis, J.-C. Fuentes-Michel, I. Weber, and M. Vossiek, "Semantic segmentation on 3d occupancy grids for automotive radar," *IEEE Access*, vol. 8, pp. 197 917–197 930, 2020.

[16] A. Piergiovanni, V. Casser, M. S. Ryoo, and A. Angelova, "4d-net for learned multi-modal alignment," in *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021, pp. 15 415–15 425.

[17] B. Yang, R. Guo, M. Liang, S. Casas, and R. Urtasun, "Radarnet: Exploiting radar for robust perception of dynamic objects," in *Computer Vision – ECCV 2020*, A. Vedaldi, H. Bischof, T. Brox, and J.-M. Frahm, Eds. Cham: Springer International Publishing, 2020, pp. 496–512.

[18] M. Shah, Z. Huang, A. Laddha, M. Langford, B. Barber, s. zhang, C. Vallespi-Gonzalez, and R. Urtasun, "Liranet: End-to-end trajectory prediction using spatio-temporal radar fusion," in *Proceedings of the 2020 Conference on Robot Learning*, ser. Proceedings of Machine Learning Research, J. Kober, F. Ramos, and C. Tomlin, Eds., vol. 155. PMLR, 16–18 Nov 2021, pp. 31–48. [Online]. Available: https://proceedings.mlr.press/v155/shah21a.html

[19] R. Nabati and H. Qi, "Centerfusion: Center-based radar and camera fusion for 3d object detection," in *2021 IEEE Winter Conference on Applications of Computer Vision (WACV)*, 2021, pp. 1526–1535.

[20] R. Weston, S. Cen, P. Newman, and I. Posner, "Probably unknown: Deep inverse sensor modelling radar," in *2019 International Conference on Robotics and Automation (ICRA)*, 2019, pp. 5446–5452.

[21] L. Sless, B. E. Shlomo, G. Cohen, and S. Oron, "Road scene understanding by occupancy grid learning from sparse radar clusters using semantic segmentation," in *2019 IEEE/CVF International Conference on Computer Vision Workshop (ICCVW)*, 2019, pp. 867–875.

[22] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie, "Feature pyramid networks for object detection," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 936–944.

[23] R. Cipolla, Y. Gal, and A. Kendall, "Multi-task learning using uncertainty to weigh losses for scene geometry and semantics," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018, pp. 7482–7491.

[24] P. Sun, Y. Jiang, E. Xie, Z. Yuan, C. Wang, and P. Luo, "Onenet: Towards end-to-end one-stage object detection," *CoRR*, vol. abs/2012.05780, 2020. [Online]. Available: https://arxiv.org/abs/2012.05780

[25] H. Caesar, V. Bankiti, A. H. Lang, S. Vora, V. E. Liong, Q. Xu, A. Krishnan, Y. Pan, G. Baldan, and O. Beijbom, "nuscenes: A multimodal dataset for autonomous driving," *arXiv preprint arXiv:1903.11027*, 2019.