Deep learning for radar data exploitation of autonomous vehicle

Arthur Ouaknine

INSTITUT POLYTECHNIQUE DE PARIS

NNT : 2022IPPAT007

Thèse de doctorat

Deep learning for radar data exploitation of autonomous vehicle

Thèse de doctorat de l'Institut Polytechnique de Paris
préparée à Télécom Paris et Valeo

École doctorale n°626 Institut Polytechnique de Paris (ED IP Paris)
Spécialité de doctorat : Signal, Images, Automatique et robotique

Thèse présentée et soutenue à Palaiseau, le 04/03/2022, par

ARTHUR OUAKNINE

Composition du Jury :

David Picard Directeur de recherche, École des Ponts ParisTech (IMAGINE)	President
Dominique Béréziat Maître de conférences, Sorbonne Université (LIP6)	Rapporteur
Francesca Bovolo Directrice de recherche, Fondazione Bruno Kessler (RSDE)	Rapporteuse
Gustau Camps-Valls Professeur, Universitat de València (IPL)	Examinateur
Guillaume Charpiat Chargé de recherche, INRIA (TAU)	Examinateur
Florence Tupin Professeure, Télécom Paris & IP Paris (LTCI)	Directrice de thèse
Patrick Pérez Directeur de recherche, Valeo	Co-directeur de thèse
Alasdair Newson Maître de conférences, Télécom Paris & IP Paris (LTCI)	Co-directeur de thèse

Acknowledgments

Je souhaite tout d'abord remercier Alasdair, Florence et Patrick, mes encadrants de thèse, pour m'avoir fait confiance sur ce projet et pour avoir dirigé mes recherches avec rigueur pendant plus de trois années. Alasdair, merci de m'avoir soutenu et aidé à tout moment, aussi bien personnellement que professionnellement ; et merci pour ton implication rigoureuse dans tous mes projets. Tu as été un acteur majeur de cette thèse sans qui elle n'aurait pas pu aller aussi loin : ne change rien pour tes futurs doctorants. Florence, merci pour ta disponibilité, pour nos discussions et nos encadrements de projets communs qui m'ont beaucoup apporté humainement et scientifiquement. Patrick, merci pour nos discussions et tes idées qui ont inspiré nos travaux ainsi que pour ta confiance en m'intégrant à l'équipe naissante de valeo.ai.

Je remercie également Julien pour avoir contribué à cette thèse de façon significative et sans qui ces travaux n'auraient pas pu voir le jour. Merci pour tes idées, ton expertise et ta disponibilité ayant permis des avancées importantes sur ces sujets que nous abordions tous les deux.

I would like to thank the jury of my thesis for your time and your relevant feedbacks which truly improved the quality of this thesis. I would like to especially thank Domique Bérézia and Francesca Bovolo for reviewing my manuscript. Je remercie aussi mon jury de mi-parcours pour leurs retours et conseils avisés concernant ma thèse.

Je remercie Télécom Paris pour son accueil et tout particulièrement l'équipe IMAGES du département IDS. J'ai eu la chance de faire de nombreuses rencontres entre Paris 13ème et le plateau de Saclay. Merci à mes co-bureaux des deux sites : Nicolas Go., Vincent, Raphaël et Matthis. Je remercie toute l'équipe IMAGES, les enseignants-chercheurs, les post-docs et les actuels et anciens doctorants pour cette vie animée à Télécom Paris. Merci à Emanuele, Mateus, Giammarco, Nicolas Ga., Clément, Antoine, Inès, Nicolas C., Robin, Xu, Xiangli, Corentin, Sylvain, Christophe, Alban, Zoé, Pietro, Saïd, Yann, Isabelle et Jean-Marc. Je remercie particulièrement Jean-Marie pour la patience dont il a fait preuve pour m'enseigner la physique élémentaire au début de ma thèse. Et merci aux co-organisateurs et aux participants du Deep Learning Working Group, qui a été une source de discussions riches et de rencontre entre les doctorants.

Je remercie également toute l'équipe de valeo.ai qui a tant évolué depuis mon arrivée. Je remercie tout particulièrement les Bezos, Charles, Maxime et Simon, pour nos débats et nos blagues. Sans eux, l'aventure n'aurait pas été la même. Merci à Gabriel de m'avoir sorti de nombreuses situations difficiles. Merci à Alexandre pour tous nos échanges très enrichissants et pour ce projet avec beaucoup d'avenir. Je remercie tous les actuels et anciens permanents, doctorants et stagiaires pour ces échanges scientifiques d'une qualité impressionnante. Merci à Andrei, Hedi, Eloi, Tuan-Hung, Himalaya, Spyros, Florent, Huy, Antoine, Oriane, Laura, Gilles, Matthieu, Renaud, Mickael, Tristan, Bjoern, Corentin, Leon et Antonin.

Je remercie tous mes anciens collègues de Zyl, a.k.a. BIM, a.k.a. Comète, a.k.a. Crossroad, pour tous ces moments incroyables, avant, pendant et après mon stage de fin d'étude. Merci à Mathieu, Aurel, Anthony, Ophély, Samy, Gauthier, Camille, Thomas, Martin, Alexandre, Flo et Valentin. Je remercie tout particulièrement Long pour m'avoir

transmis sa passion pour la recherche, pour son encadrement irréprochable et pour avoir poussé mon projet de thèse sans même le vouloir.

Je remercie tous mes amis pour faire partie de ma vie et pour m'avoir soutenu pendant toutes ces années d'études et de recherche. Merci à Gauthier, Garance et Félix, mes amis d'enfance qui ont toujours été et seront toujours là, de Paris au Guatemala. Merci aux Chocs, Toli, Matthieu, Raphaël, Simon, Mattia, Loris, Maxou et Laloum, mes frères qui ont partagé ma vie et la partageront pendant encore longtemps sdv. Merci à Lina et Anna pour m'avoir soutenu en toute condition ces dernières années, je vous en suis extrêmement reconnaissant ; notre amitié a encore de grandes choses à vivre.

Bien évidemment, je remercie Sofia, Zalie, Hélène et Fanny, les reines de Montalivet, pour votre indéfectible soutien, pour tout ce qu'on a vécu et tout ce qu'il nous reste à vivre ensemble.

Je remercie les puissant.e.s, a.k.a. grand est, pour tous ces moments incroyables de vie, de fêtes, de débats d'experts, entre Paris, Moliets et Bordeaux. Merci à Eugénie, Manon, Joachim, Romain, Solène, Hicham et Roxane, avec qui j'ai hâte de revivre ces joies.

Je remercie également Lou, Maud, Juliette, Violette, Nico, Vincent et Djav sans qui ces vacances, ces dîners, ces fêtes et autres moments n'auraient pas créé de si beaux souvenirs chers à mes yeux. Je remercie JB pour toutes ces discussions, ces apéros et ces soirées où nous avons refait le monde depuis les bancs de la Sorbonne.

Je remercie Antoine et Matthieu pour nos débats interminables, nos apéros au BM et à la Butte-aux-Cailles, depuis Télécom à aujourd'hui, vous avez réussi à me motiver pour cette thèse.

Je remercie toute ma famille, Suzanne, Christiane, Henri, Antoine, Isabelle, Didier, Michel, Mirjanna, Maud, Wahid et Jonathan pour toutes ces années de partage, qui ne cesseront jamais je l'espère.

Je remercie mes parents et ma sœur, Lydia, Robert et Lucie, pour leur amour, leur soutien, leur compréhension et leurs encouragements depuis tant d'années ; sans qui je n'aurais jamais pu arriver où je suis aujourd'hui. Cette thèse leur est dédiée.

Je remercie finalement celles et ceux que je n'ai pas cité et qui m'ont énormément apporté au cours de ma vie ; grâce à qui j'ai pu être heureux, me construire et apprendre à dépasser mes limites. Enfin, un immense merci aux lecteurs, qui auront la patience de se plonger dans mes travaux ayant occupé mes trois dernières années.

Contents

List of Acronyms vii

List of Figures x

List of Tables xiii

Abstract xv

French Summary xvii

1 Introduction 1
- 1.1 Context . . . 1
- 1.2 Motivations . . . 4
- 1.3 Contributions and outlines . . . 5

2 Background 7
- 2.1 RADAR theory . . . 7
- 2.2 Recordings and signal processing . . . 10
  - 2.2.1 Transformations in the temporal and frequency domains . . . 10
  - 2.2.2 Speckle noise . . . 12
  - 2.2.3 RADAR representations . . . 13
- 2.3 Artificial neural networks . . . 15
  - 2.3.1 Introduction . . . 15
  - 2.3.2 At the neuron level . . . 16
  - 2.3.3 At the layer level . . . 17
  - 2.3.4 Training a neural network . . . 18
- 2.4 Convolutional neural network . . . 22
  - 2.4.1 Convolutional layer . . . 22
  - 2.4.2 Complementary methods and layers . . . 22
- 2.5 Recurrent neural network . . . 24
- 2.6 Deep learning . . . 26
  - 2.6.1 Classification . . . 26
  - 2.6.2 Object detection . . . 29
  - 2.6.3 Semantic segmentation . . . 33
  - 2.6.4 Methods for 3D point clouds . . . 38

3 Related work 41
- 3.1 Diverse applications . . . 41
- 3.2 Automotive RADAR datasets . . . 43
  - 3.2.1 Traditional RADAR . . . 44
  - 3.2.2 Scanning RADAR . . . 45

3.2.3 High-definition RADAR . . . 45
3.2.4 Our proposals . . . 46
3.3 RADAR object detection . . . 46
3.3.1 Range-Angle-Doppler tensor . . . 47
3.3.2 Range-Angle or Range-Doppler view . . . 47
3.3.3 RADAR point cloud . . . 48
3.4 RADAR semantic segmentation . . . 49
3.4.1 Range-Angle view . . . 50
3.4.2 RADAR point cloud . . . 50
3.5 Sensor fusion . . . 51
3.5.1 RADAR and camera fusion . . . 51
3.5.2 RADAR and LiDAR fusion . . . 53
3.5.3 RADAR, camera and LiDAR fusion . . . 55
3.6 Conclusions . . . 55

4 Proposed automotive RADAR datasets 57
4.1 RADAR simulation . . . 58
4.1.1 Parameters and properties . . . 58
4.1.2 RadarSim dataset . . . 59
4.1.3 Experiments and results . . . 61
4.1.4 Discussions . . . 62
4.2 RADAR data generation . . . 63
4.2.1 Dataset . . . 63
4.2.2 Range-Doppler representation . . . 64
4.2.3 Methods and Experiments . . . 66
4.2.4 Discussions . . . 70
4.3 CARRADA dataset . . . 70
4.3.1 Dataset . . . 71
4.3.2 Pipeline for annotation generation . . . 72
4.3.3 Semantic segmentation baseline . . . 79
4.3.4 Discussions . . . 81
4.3.5 Conclusions . . . 81
4.4 Conclusions . . . 82

5 RADAR scene understanding 85
5.1 Multi-view RADAR semantic segmentation . . . 86
5.1.1 Motivations . . . 86
5.1.2 Methods and architectures . . . 86
5.1.3 Experiments on the CARRADA dataset . . . 92
5.1.4 Experiments on complex urban scenes datasets . . . 96
5.1.5 Conclusions and perspectives . . . 100
5.2 Sensor fusion . . . 101
5.2.1 Introduction . . . 101
5.2.2 Method . . . 102

5.2.3 Simulation . . . 106
5.2.4 Application to the nuScenes dataset . . . 106
5.2.5 Discussions and future work . . . 108
5.3 Conclusions . . . 109

6 High-definition RADAR 111
6.1 Motivations . . . 112
6.2 RADIal dataset . . . 113
6.3 Proposed method . . . 115
6.3.1 MIMO pre-encoder . . . 115
6.3.2 FPN encoder . . . 117
6.3.3 Range-Angle decoder . . . 117
6.3.4 Multi-task learning . . . 117
6.4 Experiments and Results . . . 119
6.4.1 Training details . . . 119
6.4.2 Baselines . . . 119
6.4.3 Evaluation metric . . . 120
6.4.4 Performance analysis . . . 120
6.4.5 Complexity analysis . . . 121
6.5 Conclusions and discussions . . . 122

7 Conclusion 125
7.1 Contributions . . . 125
7.2 Future work . . . 127

Bibliography 131

Appendices 155

A Background 157
A.1 Deep learning . . . 157
A.1.1 Object detection . . . 157
A.1.2 Segmentation . . . 158

B RADAR scene understanding 161
B.1 Multi-view RADAR semantic segmentation . . . 161
B.1.1 RAD tensor visualisation . . . 161
B.1.2 Detailed multi-view architectures . . . 161
B.1.3 Pre-processing and training procedures . . . 162
B.1.4 Quantitative results . . . 162
B.1.5 Variability of the quantitative results by method . . . 162
B.1.6 Variability of the loss ablation study . . . 165
B.1.7 Qualitative results on CARRADA . . . 166
B.1.8 Qualitative results on RADDet . . . 167

B.1.9 Qualitative results on in-house dataset 167
B.2 Sensor fusion 168
B.2.1 Sensor settings of the nuScenes dataset 168
B.2.2 Qualitative results of the propagation and fusion module 169

C High-definition RADAR **173**
C.1 Ablation study of the MIMO pre-encoder 173

List of Acronyms

AD Angle-Doppler . . . 13

ADAS Advanced Driving Assistance Systems . . . 1

ADC Analog-to-Digital Converter . . . 15

AE AutoEncoder . . . 48

AN Artificial Neuron . . . 16

ANN Artificial Neural Network . . . 3

AoA Angle-of-Arrival . . . 9

AP Average Precision . . . 30

AR Average Recall . . . 35

ASPP Atrous Spatial Pyramidal Pooling . . . 37

BCE Binary Cross-Entropy . . . 18

BEV Bird's Eye View . . . 38

CAN bus Controller Area Network . . . 113

CE Cross-Entropy . . . 18

CFAR Constant False Alarm Rate . . . 14

cGAN conditional Generative Adversarial Network . . . 53

CNN Convolutional Neural Network . . . 4

CoL Coherence Loss . . . 91

DFT Discrete Fourier Transform . . . 10

DoA Direction-of-Arrival . . . 14

FCN Fully Convolutional Network . . . 35

FFT Fast Fourier Transform . . . 10

FMCW Frequency-Modulated Continuous Wave . . . 8

FoV Field-of-View . . . 45

FPN Feature Pyramid Network . . . 48

FPS Frames-per-Second 32

GAN Generative Adversarial Network 43

GFLOPS Giga FLoating-point Operations Per Second 112

GPS Global Positioning System 113

GPU Graphics Processing Unit 4

HD High-Definition xv

HNM Hard Negative Mining 32

IF Intermediate Frequency 11

iid independent and identically distributed 12

ILSVRC ImageNet Large Scale Visual Recognition Challenge 4

IoU Intersection over Union 30

JS Jensen-Shannon 77

KL Kullback-Leibler 77

LD Low-Definition 9

LeakyReLU Leaky Rectified Linear Unit 17

LiDAR Light Detection And Ranging xvii

LSTM Long Short-Term Memory 26

MAE Mean Absolute Error 66

mAP mean Average Precision 30

mAR mean Average Recall 35

mDice mean Dice 92

MIMO Multiple Input Multiple Output 8

mIoU mean Intersection over Union 34

MLP Multilayer Perceptron 3

MSE Mean Squared Error 18

NMS Non-Maximum Suppression 32

PP Pixel Precision 80

PR Pixel Recall 80

RA Range-Angle 13

RAD Range-Angle-Doppler 11

RAED Range-Azimuth-Elevation-Doppler 112

RCS Radar Cross Section 14

RD Range-Doppler 13

RADAR Radio Detection And Ranging xvii

ReLU Rectified Linear Unit 17

RGB Red, Green and Blue 22

RNN Recurrent Neural Network 7

RoI Region of Interest 31

RPN Region Proposal Network 31

Rx Receiver antenna 8

SAR Synthetic Aperture RADAR 43

SDice Soft Dice 90

SGD Stochastic Gradient Descent 19

SORT Simple and Online Real time Tracking 72

Tx Transmitter antenna 7

wMAE weighted Mean Absolute Error 67

wCE weighted Cross-Entropy 90

List of Figures

1.1 Sensor setup of the Valeo Drive4U prototype . . . 2
1.2 Timeline of the evolution of Artificial Neural Networks and deep learning . 3
1.3 Camera image examples of scenes with different lighting and weather conditions . . . 5

2.1 Example of a RADAR chirp . . . 8
2.2 Range-Angle-Doppler (RAD) tensor generation . . . 10
2.3 Outline of the RADAR pipeline . . . 12
2.4 An example of RADAR representations . . . 14
2.5 Illustration of a biological neuron and its mathematical model . . . 16
2.6 Illustration of a Multilayer Perceptron . . . 18
2.7 Example of Dropout . . . 21
2.8 Example of a convolutional layer . . . 23
2.9 Example of a max-pooling layer . . . 24
2.10 Example of a recurrent neural network . . . 25
2.11 Comparison between different computer vision applications . . . 27

3.1 Examples of Low-Definition, High-Definition and Scanning RADARs . . . 45
3.2 Example of the annotation of a 'Car' signature in Range-Doppler . . . 49

4.1 Example of a Range-Doppler simulation for the category 'Car' . . . 60
4.2 Example of two scenes with pedestrians and moving cars . . . 64
4.3 Distributions of the background in RADAR data . . . 65
4.4 Qualitative results of Range-Doppler generations . . . 70
4.5 A scene from CARRADA dataset, with a pedestrian and a car . . . 71
4.6 Object distribution across CARRADA . . . 73
4.7 Distribution of radial velocities for all categories . . . 73
4.8 Estimation of the radial velocity from natural images . . . 74
4.9 Tracking of the Mean Shift cluster to propagate the annotation in the sequence 75
4.10 Comparison of bandwidth selection methods . . . 78
4.11 Comparison of methods for dense mask annotation generation . . . 79
4.12 Two scenes from CARRADA dataset, one with a car, the other on with a cyclist and a car . . . 82

5.1 Overview of our multi-view approach to semantic segmentation of RADAR signal . . . 86
5.2 Multi-view architectures for RADAR semantic segmentation . . . 88
5.3 Computation of the coherence loss . . . 91
5.4 Performance-*vs*.-complexity plots for all methods in Range-Doppler (RD) and Range-Angle (RA) tasks . . . 95

5.5 Qualitative results on a test scene of CARRADA . . . 97
5.6 RADDet dataset distributions . . . 98
5.7 Qualitative results on a test scene of RADDet . . . 100
5.8 Overview of our propagation and fusion approach for RADAR and LiDAR point clouds . . . 101
5.9 Simulation of the LiDAR and RADAR point cloud propagation and fusion method . . . 107
5.10 Qualitative results on the nuScenes dataset of our propose propagation and fusion module . . . 109

6.1 Overview of our RADIal dataset . . . 113
6.2 Overview of FFT-RadNet . . . 115
6.3 Trainable MIMO pre-encoder . . . 116
6.4 Qualitative results for object detection and free space segmentation on Easy and Hard samples . . . 121

B.1 Visualisation of the Range-Angle-Doppler (RAD) tensor . . . 161
B.2 Performance-*vs*.-complexity plots for all methods in RD and RA tasks . . . 166
B.3 Variability of the performances-*vs*.-complexity plots for all methods in Range-Doppler and Range-Angle tasks regarding the mDice metric. . . . 167
B.4 Variability of the performances-*vs*.-complexity plots for all methods in Range-Doppler and Range-Angle tasks regarding the mIoU metric. . . . 168
B.5 Variability of the performances plots for all combination of losses in Range-Doppler and Range-Angle tasks regarding the mDice metric. . . . 169
B.6 Qualitative results on two test scenes of CARRADA-Test . . . 170
B.7 Qualitative results on a test scene of RADDet . . . 171
B.8 Qualitative results on two test scenes of in-house sequences . . . 171
B.9 Qualitative results on two test scenes of in-house sequences . . . 172
B.10 Qualitative results on the nuScenes dataset of our propose propagation and fusion module . . . 172

C.1 MIMO pre-encoder ablation study . . . 173

List of Tables

3.1 Publicly-available driving RADAR datasets 44

4.1 Range of values for the simulated RADAR points properties 59
4.2 Performances on the RadarSim-Val and -Test datasets 62
4.3 Quantitative performances of Range-Doppler reconstruction 69
4.4 Parameters and settings of the RADAR sensor 72
4.5 Statistics of the CARRADA dataset 72
4.6 Semantic segmentation performances (%) on the test dataset for Range-Doppler (RD) and Range-Angle (RA) representations 80

5.1 Semantic segmentation performance on the CARRADA-Test dataset for Range-Doppler (RD) and Range-Angle (RA) views) 94
5.2 Ablation study of our proposed architectures 96
5.3 Ablation study of the combination of losses 96
5.4 Semantic segmentation performance on the RADDet-Test dataset for Range-Doppler and Range-Angle views 99
5.5 Parameters and settings of the RADAR sensors used in the nuScenes dataset 108

6.1 Specifications of the RADIal's sensor suite 114
6.2 Scene-type proportions in RADIal 114
6.3 Object detection performances on RADIal Test split 119
6.4 Free driving space segmentation performances 122
6.5 Complexity analysis of FFT-RadNet 122

B.1 Multi-view network (MV-Net) architecture 163
B.2 Temporal multi-view network with ASPP modules (TMVA-Net) architecture 164
B.3 Hyper-parameters used for training 165

Abstract

Autonomous driving requires a detailed understanding of complex driving scenes. The redundancy and complementarity of the vehicle's sensors provide an accurate and robust comprehension of the environment, thereby increasing the level of performance and safety. This thesis focuses the on automotive RADAR, which is a low-cost active sensor measuring properties of surrounding objects, including their relative speed, and has the key advantage of not being impacted by adverse weather conditions.

With the rapid progress of deep learning and the availability of public driving datasets, the perception ability of vision-based driving systems (e.g., detection of objects or trajectory prediction) has considerably improved. The RADAR sensor is seldom used for scene understanding due to its poor angular resolution, the size, noise, and complexity of raw RADAR data as well as the lack of available datasets. This thesis proposes an extensive study of RADAR scene understanding, from the construction of an annotated dataset to the conception of adapted deep learning architectures.

First, this thesis details approaches to tackle the current lack of data. A simple simulation as well as generative methods for creating annotated data are presented. It also describes the CARRADA dataset, composed of synchronised camera and RADAR data with a semi-automatic method generating annotations on the RADAR representations. Today, the CARRADA dataset is the only one to provide annotated raw RADAR data for object detection and semantic segmentation tasks. The CARRADA dataset and its annotation method form our first major contribution presented at the International Conference on Pattern Recognition (ICPR).

This thesis then presents a proposed set of deep learning architectures for RADAR semantic segmentation. A combination of specialized loss functions is also presented, including a coherence function that reinforces the spatial harmony of the predictions made by the model on the different views of the RADAR tensor. The proposed architecture with the best results outperforms alternative models, derived either from semantic segmentation of natural images or from RADAR scene understanding, while requiring much less parameters. The presented method for semantic segmentation of raw RADAR data is our second major contribution presented at the International Conference on Computer Vision (ICCV). This thesis also introduces a method to open up research into the fusion of LiDAR and RADAR sensors for scene understanding. The proposed method is an early fusion module propagating the RADAR information in the LiDAR point cloud by considering the resolution and accuracy of the sensor as a quantification of its uncertainty. In this way, our module is able to create a point cloud that benefits from the advantages of both sensors while compensating for their disadvantages.

Finally, this thesis exposes a collaborative contribution, the RADIal dataset with synchronised High-Definition (HD) RADAR, LiDAR and camera. A deep learning architecture is also proposed to estimate the RADAR signal processing pipeline while performing multitask learning for object detection and free driving space segmentation simultaneously. This collaborative contribution is under review at an international conference.

French Summary

La conduite autonome vise à révolutionner notre mobilité en comprenant et en prévoyant l'environnement de conduite tout en palliant les faiblesses humaines de manière sûre et efficace. Pour se faire, elle exige une compréhension détaillée de scènes de conduite complexes. L'environnement du véhicule est observé par des caméras enregistrant des images facilement compréhensibles par l'œil humain. Au fil du temps, des capteurs actifs supplémentaires sont apparus, venant compléter les caméras et permettant une meilleure compréhension des scènes environnantes : le Light Detection And Ranging (LiDAR), le Radio Detection And Ranging (RADAR) et les capteurs à ultrasons. La redondance et la complémentarité des capteurs du véhicule permettent ainsi une compréhension précise et robuste de l'environnement : ils augmentant ainsi le niveau de performance et de sécurité. Cette thèse se concentre sur le RADAR automobile, qui est un capteur actif à faible coût mesurant les propriétés des objets environnants, y compris leur vitesse relative, tout en ayant l'avantage de ne pas être affecté par des conditions météorologiques défavorables.

Avec les progrès rapides de l'apprentissage profond et la disponibilité d'ensembles de données publiques sur la conduite, la capacité de perception des systèmes de conduite basés sur la vision (par exemple, la détection d'objets ou la prédiction de trajectoire) s'est considérablement améliorée. Le capteur RADAR est rarement utilisé pour la compréhension de scène en raison de sa faible résolution angulaire, de la taille, du bruit et de la complexité des données brutes RADAR ainsi que du manque de données disponibles. Cette thèse propose une étude approfondie de la compréhension de scènes RADAR : de la construction d'un jeu de données annotées à la conception d'architectures d'apprentissage profond adaptées.

Tout d'abord, le chapitre 1 de cette thèse introduit les motivations ayant conduit à nos travaux. En particulier, il aborde les grandes avancées de la conduite autonome ainsi que l'avènement des algorithmes d'apprentissage.

Le chapitre 2 présente ensuite la théorie de fonctionnement du capteur RADAR et les méthodes de traitement du signal appliquées aux données enregistrées. Il présente également la théorie des réseaux de neurones artificiels ainsi que les architectures d'apprentissage profond pour l'analyse d'images qui sont, pour la plupart, exploitées dans nos travaux.

Le chapitre 3 présente les ensembles de données RADAR existants et les travaux antérieurs sur la compréhension de scènes RADAR pour la détection d'objets et la segmentation sémantique basées sur les algorithmes d'apprentissage profond. Il traite aussi des approches antérieures de fusion de capteurs combinant le RADAR avec des capteurs de type caméra ou LiDAR, ou les deux, pour la compréhension de scènes automobiles.

Le chapitre 4 de cette thèse détaille ensuite des approches permettant de remédier au manque de données. Une simulation simple ainsi que des méthodes génératives pour créer des données annotées sont présentées. Ce chapitre décrit également le jeu de données CARRADA, composé de données synchronisées de caméra et de RADAR avec une méthode semi-automatique générant des annotations sur les représentations RADAR. Aujourd'hui, le jeu de données CARRADA est le seul à fournir des données RADAR brutes annotées pour des tâches de détection d'objets et de segmentation sémantique. Le jeu de

données CARRADA et sa méthode d'annotation forment notre première contribution majeure présentée à la conférence internationale sur la reconnaissance de forme (International Conference on Pattern Recognition).

Le chapitre 5 présente ensuite un ensemble d'architectures d'apprentissage profond pour la segmentation sémantique de données RADAR brutes. Une combinaison de fonctions de perte spécialisées est également proposée dont une fonction de cohérence renforçant l'harmonie spatiale des prédictions réalisées par le modèle sur les différentes vues du tenseur RADAR. L'architecture proposée la plus performante présente les meilleures performances comparées aux modèles alternatifs, dérivés soit de la segmentation sémantique d'images naturelles, soit de la compréhension de scènes RADAR, tout en nécessitant beaucoup moins de paramètres. La méthode de segmentation sémantique de données RADAR brutes présentée est notre seconde contribution majeure présentée à la conférence internationale de vision par ordinateur (International Conference on Computer Vision). Ce chapitre décrit également une méthode permettant d'ouvrir la recherche sur la fusion des capteurs RADAR et LiDAR pour la compréhension de scènes. La méthode proposée est composée d'un module de fusion précoce propageant l'information RADAR dans le nuage de points LiDAR en considérant la résolution et la précision du capteur RADAR comme une quantification de son incertitude. De cette manière, notre méthode est capable de créer un nuage de points bénéficiant des atouts des deux capteurs tout en palliant leurs inconvénients.

Le chapitre 6 expose une contribution collaborative, le jeu de données RADIal enregistré avec un RADAR haute définition (HD), un LiDAR et une caméra synchronisés. Une architecture d'apprentissage profond est également proposée pour estimer la chaîne de traitement du signal RADAR tout en effectuant simultanément un apprentissage multitâche pour la détection d'objets et la segmentation de l'espace libre de conduite. Cette contribution collaborative est en cours de révision dans une conférence internationale.

Enfin, le chapitre 7 de cette thèse conclue sur nos contributions et les limites de nos travaux. Il présente aussi nos futurs travaux sur la compréhension de scènes RADAR. En particulier, les pistes d'amélioration de notre méthode de segmentation sémantique de données RADAR brutes sont abordées. Ce chapitre aborde aussi nos perspectives d'exploitation de nuages de points RADAR et LiDAR propagés et fusionnés dans le but d'obtenir une meilleure compréhension des scènes urbaines complexes.

CHAPTER 1

Introduction

Contents

1.1 Context 1
1.2 Motivations 4
1.3 Contributions and outlines 5

1.1 Context

Autonomous driving aims to revolutionize our mobility by understanding and predicting the driving environment while alleviating for human weaknesses in a safe and efficient manner. Since the European project Eureka Prometheus (1987-1995) and the American DRAPA Grand Challenge (2004-2007), the spectacular progresses in visual scene understanding using learning algorithms have brought back the old dream of driverless cars [Janai *et al.* 2020]. The automotive industry is getting closer and closer to this goal as the major players have massively invested in advanced technologies: multi-sensor perception, three-dimensional reconstruction, cartography, high-precision location, planning and commands.

Scene understanding is a prerequisite, but it goes hand in hand with decision making for car automation. Both must be mastered to deliver safe vehicles with automatic features to the public. The automotive industry distinguishes five levels of driving automation. Levels one and two consist of integrating driving assistance or partially automatic features, while a human continues to monitor all tasks, *e.g.* automatic cruise control or lane centering. These levels are already integrated in the vehicles currently marketed to the public. At level three, the vehicle is capable of performing most driving tasks in certain conditions, *e.g.* highway driving up to 60km/h, but human control is still required for these features. The automotive industry estimates that level 3 vehicles will be sold to the public in 2024 in European countries. Levels four and five no longer require a driver and perform fully automatic driving in different circumstances. Depending on the progress of research and development, these levels could become publicly available in the coming decades.

Advanced Driving Assistance Systems (ADAS) and autonomous driving require a detailed understanding of complex driving scenes. The environment of the vehicle is observed by cameras recording images that are easily understood by the human eye. Additional active sensors have emerged complementing cameras and allowing a better understanding of the surrounding scenes: Light Detection And Ranging (LiDAR), Radio Detection And

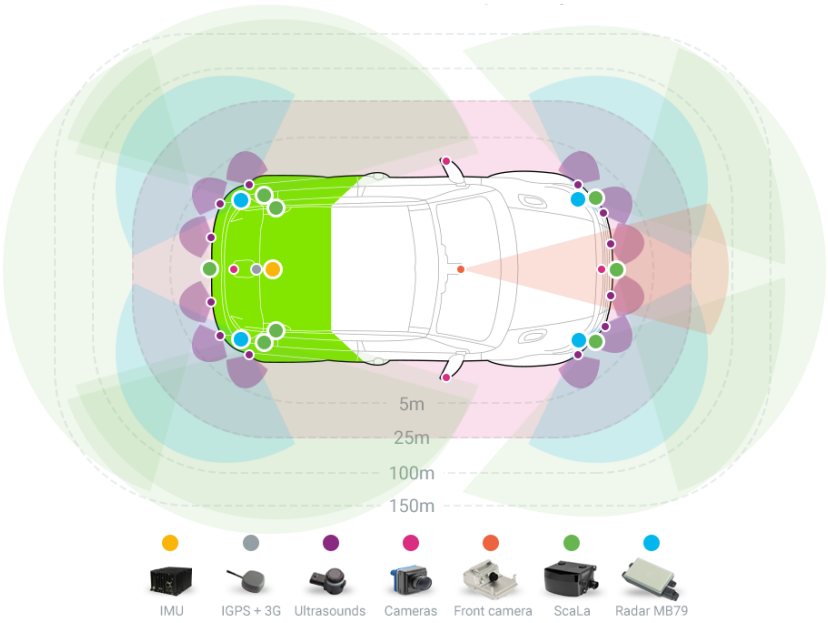


Figure 1.1: **Sensor setup of the Valeo Drive4U prototype**. Source: Valeo.

Ranging (RADAR) and ultrasound sensors. With the transition from assisted to automated driving, the redundancy and complementary of these sensors aims to provide an accurate and robust comprehension of the environment, thereby increasing the level of performance and safety. Redundancy mechanisms are required at all levels of the system: from the sensing parts to the final decision modules. At the sensor level, they can be reached using sensors of different types, where each of them offers a vision of the world with its own properties and physical aspects.

The automotive supplier Valeo is a world leader in the design and manufacturing of sensors for assistance driving. The company has been actively involved in autonomous driving research and development for almost ten years, designing sensor prototypes and real-world experiments (see Figure 1.1). In this context, prototype cars are equipped with various sensors to record real-life scenes and to understand the complementarity and limitations between the sensors. The recorded and released data from the automotive industry in collaboration with researchers aim developing research on recent learning algorithms to better understand the car's environment.

This thesis focuses on automotive RADARs which are low-cost active sensors measuring properties of surrounding objects, including their relative speed, and have the key advantage of not being impacted by adverse weather conditions, *e.g.* rain, snow or fog. They have been used in the automotive industry for the last two decades, *e.g.*, for automatic cruise control or blind spot detection. RADAR has become the sensor of choice for applications requiring time to collision as it provides, besides localization, the radial velocity thanks to the Doppler information. However, it is seldom used for scene understanding due to its poor angular resolution, the size, noise, and complexity of RADAR raw data as

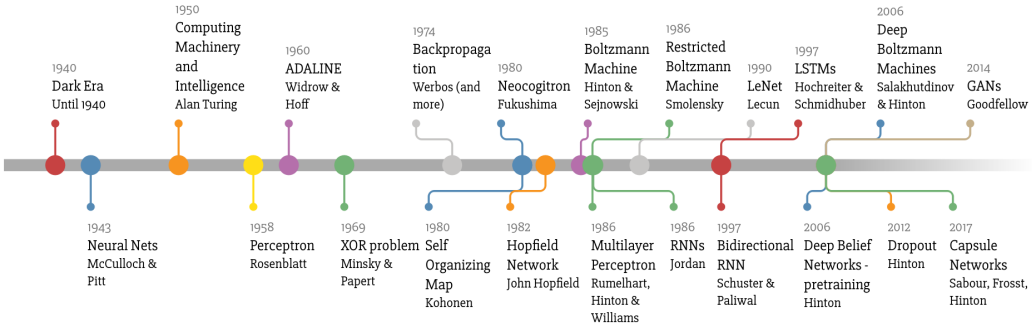


Figure 1.2: **Timeline of the evolution of Artificial Neural Networks and deep learning**[2].

well as the lack of available datasets. Representations provided by a RADAR sensor are moreover difficult to understand for non-specialist humans. For these reasons, RADAR sensors were left out of scene understanding using learning algorithms. This thesis aims to exploit RADAR data for object recognition in complex driving scenes using modern machine learning approaches.

Machine learning, as part of artificial intelligence, is the study of computer algorithms that automatically improve their performance by experience. These algorithms use a dataset, or training data, to learn to recognize patterns and make predictions. Machine learning algorithms have rapidly developed since the advent of the digital economy and the abundance of available data. They are applied in many fields, *e.g.* to predict consumer behaviour, estimate financial time series, detect cancer cells in medical images, forecast weather, rank user preferences in video platforms or understand scenes and take decisions in autonomous driving.

This thesis is centered on a specific category of machine learning algorithms called Artificial Neural Networks (ANNs). In the human brain, a neuron receives electrical signals via *dendrites*. The signals are then processed in the *soma*, transmitted by the *axon* if the signal is above a threshold, and transferred to the next neuron with the *synapses*. Inspired by this phenomenon, [McCulloch & Pitts 1943] proposed the first mathematical formulation of an artificial neuron processing a signal with fixed weights (similarly to *dendrites*) and applying a threshold on the output. An extension proposed by [Rosenblatt 1958] updates the weights regarding the prediction error of the artificial neuron. This algorithm called the Perceptron is the first biologically inspired machine learning algorithm from the human brain. In the late 1960's, [Minsky & Papert 1969] showed the limitations of a single artificial neuron which is a linear classifier and therefore cannot solve non-linearly separable classification problems (*e.g.* XOR). They proposed to overcome this problem by stacking artificial neurons in layers and by connecting the neurons from a layer to another creating a neural network, also called Multilayer Perceptron (MLP).

Over the years, several improvements have been published, as illustrated in Figure

[2] https://towardsdatascience.com/a-weird-introduction-to-deep-learning-7828803693b0

1.2, proposing neural network structures or innovating methods to train these algorithms. An important step forward is the Convolutional Neural Network (CNN), introduced by [LeCun *et al.* 1989] associated with a new learning method, which is particularly adapted to image analysis. However, it was hampered by the computer limitations of the time. In the 2010's, the increase of computing power, in particular with Graphics Processing Units (GPUs), and the availability of large-scale databases motivated the emergence of deep learning. Deep learning is a subset of machine learning that involves stacking many layers in a neural networks. These algorithms were highlighted with a major milestone reached in 2012 with the ImageNet Large Scale Visual Recognition Challenge (ILSVRC), consisting of classifying images into 1,000 categories (*e.g.* cats, dogs, or food). The first deep learning model proposed by [Krizhevsky *et al.* 2012] outperformed all the existing methods classifying image in a category with a large margin. In 2015, another deep learning model introduced by [He *et al.* 2016] first beat human performance on the ImageNet challenge making deep learning methods the front page of the computer vision research community.

Since the 2010's, deep neural networks have outperformed humans in face recognition [Taigman *et al.* 2014] or in the game of Go with AlphaGo [Silver *et al.* 2016]. The ANN have been adapted to many types of data: images, videos, point clouds, texts, sounds, time series or DNA sequences. These impressive performances have attracted the automotive industry, which sees these algorithms as an answer to the fundamental requirements for autonomous driving. They are particularly well suited for scene understanding to detect and classify objects in a car's environment using camera or LiDAR data. This thesis aims to investigate deep learning methods for RADAR data, which have barely been explored in recent years.

1.2 Motivations

Scene understanding requires a high level of perception around the car. For this purpose, complementary sensors are used to compensate their respective weaknesses. Cameras record images that provide a comprehensive understanding of a scene, but they are impacted by lighting conditions, *e.g.* during night or when facing the sun, or by adverse weather conditions that reduce their visibility (see Figure 1.3). LiDAR is an active sensor transmitting laser beams in the environment and is not affected by lighting conditions. This sensor measures the time it takes to receive reflected light and records the distance and the intensity of a reflection. The LiDAR sensor provides a 3D point cloud measuring the geometry of a scene with a resolution below the degree for both azimuth and elevation angles. However, the laser beams are reflected by droplets or snowflakes due to their wavelength. Difficult weather conditions create artefacts in the recorded 3D point clouds *e.g.* with rain [Karlsson *et al.* 2021], snow [Kurup & Bos 2021] or fog [Bijelic *et al.* 2018, Guan *et al.* 2020]. The RADAR sensor emits electromagnetic waves, which are not impacted by adverse weather conditions, and records the location, Doppler and reflectivity of objects in the scenes through the received signals. This sensor is an interesting candidate for scene understanding because its properties compensate for other

Figure 1.3: **Camera image examples of scenes with different lighting and weather conditions**.

sensors in specific scenarios.

With the rapid progress of deep learning and the availability of public driving datasets, *e.g.*, [Geiger *et al.* 2013, Cordts *et al.* 2016, Huang *et al.* 2019, Yu *et al.* 2020, Sun *et al.* 2020], the perception ability of vision-based driving systems (detection of objects, structures, markings and signs, estimation of depth, forecasting of other road users' movements) has considerably improved. This progress quickly extended to LiDAR, with the help of specific architectures to deal with 3D point clouds [Qi *et al.* 2017a]. To improve further the performance of autonomous driving systems, extending the size and scope of open annotated datasets is a key challenge. The RADAR sensor provides cumbersome and noisy representations which are difficult to understand. For these reasons, there was no open source RADAR dataset for automotive application before 2019, which has hampered research on deep learning applied to RADAR data. Motivated by the relevance of this sensor, this thesis fills the gap in existing works by proposing unique datasets and methods to generate annotations avoiding costly manual annotations.

While deep learning has brought major progresses in the automotive use of cameras and LiDARs – for object detection and segmentation in particular – it is only recently that it has also embraced RADAR signals. In fact, even though the RADAR technology has greatly improved, the signal processing pipeline has remained the same for years. This sensor is now a source of interest since public datasets have been released as detailed in Section 3.2. Motivated by the recent advances in deep learning algorithms, this thesis proposes to adapt neural network architectures for RADAR scene understanding.

1.3 Contributions and outlines

This thesis proposes an extensive analysis of RADAR scene understanding, from the construction of an annotated dataset to the conception of adapted deep learning architectures.

Chapter 2 provides the background on RADAR theory and signal processing methods applied to the recorded data. It also briefly introduces the theory behind ANN as well as present deep learning architectures based on images which are exploited in our work.

Chapter 3 relates the existing RADAR datasets and previous works on RADAR scene understanding for object detection and semantic segmentation using deep learning algorithms. It also presents previous sensor fusion approaches combining RADAR with either or both camera and LiDAR sensors for automotive scene understanding.

Chapter 4 details approaches to tackle the lack of data. A simple simulation as well

as generative methods for creating annotated data are presented. This chapter also introduces one major contribution, the CARRADA dataset, composed of synchronised camera and RADAR data with a semi-automatic method generating annotations on the RADAR representations. Today, the CARRADA dataset is the only dataset providing annotated raw RADAR data for object detection and semantic segmentation tasks.

Chapter 5 presents our second major contribution, a set of deep learning architectures with their associated loss functions for RADAR semantic segmentation. Today, the proposed architecture performing the best outperforms alternative models, derived either from the semantic segmentation of natural images or from RADAR scene understanding, while requiring significantly fewer parameters. This chapter also introduces a work in progress to open up research into the fusion of LiDAR and RADAR sensors for scene understanding.

Chapter 6 exposes a collaborative contribution, the RADIal dataset with synchronised High-Definition (HD) RADAR, LiDAR and camera. A deep learning architecture is also proposed to estimate the RADAR signal processing pipeline while performing multi-task learning for object detection and free driving space segmentation[3] simultaneously. Today, RADIal is the only dataset providing raw data from a HD RADAR with annotations for both object detection and free space segmentation. Moreover, the proposed method is the only one to successfully learn a part of the RADAR pre-processing pipeline in a multi-task framework using HD RADAR data.

Finally, Chapter 7 highlights the contributions of this thesis and discuss the perspectives of exploiting RADAR data for scene understanding in the context of autonomous driving.

The contributions presented in this thesis have led to the following publications:

Publication	Chapter
Arthur Ouaknine, Alasdair Newson, Julien Rebut, Florence Tupin and Patrick Pérez. "CARRADA Dataset: Camera and Automotive Radar with Range-Angle-Doppler Annotations", in *International Conference on Pattern Recognition (ICPR)*, 2020.	4
Arthur Ouaknine, Alasdair Newson, Patrick Pérez, Florence Tupin and Julien Rebut. "Multi-View Radar Semantic Segmentation", in *International Conference on Computer Vision (ICCV)*, 2021.	5
Julien Rebut, **Arthur Ouaknine**, Waqas Malik and Patrick Pérez. "Raw High-Definition Radar for Multi-Task Learning", in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.	6

[3]The free driving space segmentation task consists to locate pixel-wise the available space that can be driven.

CHAPTER 2

Background

Contents

2.1 RADAR theory 7
2.2 Recordings and signal processing 10
2.2.1 Transformations in the temporal and frequency domains 10
2.2.2 Speckle noise 12
2.2.3 RADAR representations 13
2.3 Artificial neural networks 15
2.3.1 Introduction 15
2.3.2 At the neuron level 16
2.3.3 At the layer level 17
2.3.4 Training a neural network 18
2.4 Convolutional neural network 22
2.4.1 Convolutional layer 22
2.4.2 Complementary methods and layers 22
2.5 Recurrent neural network 24
2.6 Deep learning 26
2.6.1 Classification 26
2.6.2 Object detection 29
2.6.3 Semantic segmentation 33
2.6.4 Methods for 3D point clouds 38

This chapter provides the background to understand the automotive RADAR sensor and the neural network machine learning algorithms. Section 2.1 presents the theory of the automotive RADAR sensor. Section 2.2 details the pipeline commonly used to process the recorded RADAR signals. Sections 2.3, 2.4 and 2.5 introduce the ANN, Recurrent Neural Network (RNN) and Convolutional Neural Network (CNN) respectively. Finally, Section 2.6 relates deep learning methods and architectures which are exploited in the following chapters.

2.1 RADAR theory

A Radio Detection And Ranging (RADAR) sensor is an active sensor using its own source of signal. It generates electromagnetic waves which are emitted via one or several Transmitter antennas (Txs). The wavelengths used are significantly larger than the

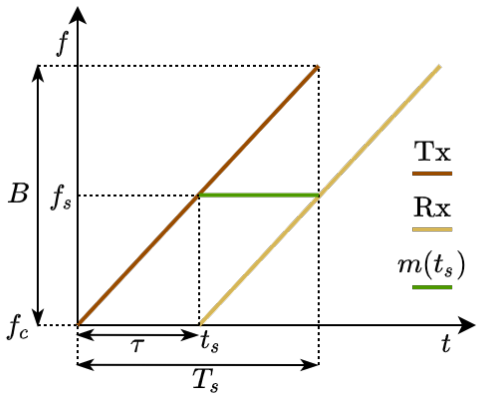


Figure 2.1: **Example of a RADAR chirp**. The chirp is generated considering a linearly modulated frequency as described in Equation 2.1. The signal is transmitted by a Tx antenna and received by a Rx antenna for each reflector in the environment. The IF signal is deduced from a mixer by comparing the transmitted and received signals (see Equation 2.12). A signal reflected by a single object is illustrated here. Multiple signals are separated regarding the time delay of the signal round-trips and the phase shift of the waves. It would be illustrated by distinct IFs, one per object.

visible spectrum and therefore not affected by lighting or adverse weather conditions. The waves are reflected by an object depending on the composition and the geometry of its surface. The signals are then received by the RADAR via one or several Receiver antennas (Rxs). The comparison between the transmitted and the received waveforms allows inferring the distance, the relative velocity, the azimuth angle and the elevation of the reflector regarding the RADAR position [Ghaleb 2009] and the positioning of its antennas. Most of the automotive RADARs use Multiple Input Multiple Output (MIMO) systems [Donnet & Longstaff 2006]: each couple of (Tx, Rx) receives the reflected signal assigned to a specific Tx transmitting a waveform.

A Frequency-Modulated Continuous Wave (FMCW) RADAR transmits a signal, called a chirp [Brooker 2005], whose frequency is linearly modulated over the sweeping period T_s: at time $t_s \in \{0, \cdots, T_s\}$, the emitted sinusoidal signal has the frequency:

$$f_s = f_c + \frac{B}{T_s} t_s, \tag{2.1}$$

where f_c is the carrier frequency, B the bandwidth, B/T_s is the linear slope of the frequency variation, and its phase reads

$$\phi_E(t) = 2\pi f_s t. \tag{2.2}$$

After reflection on an object at distance $r(t)$ from the emitter, the received signal has phase:

$$\phi_R(t) = 2\pi f_s (t - \tau) = \phi_E(t) - \phi(t), \tag{2.3}$$

where $\tau = \frac{2r(t)}{c}$ is the time delay of the signal round trip, with c the velocity of the wave through the air considered as constant, and $\phi(t)$ is the phase shift:

$$\phi(t) = 2\pi f_s \tau = 2\pi f_s \frac{2r(t)}{c}. \tag{2.4}$$

Measuring this phase shift (or equivalently the time delay between the transmitted and the reflected signal) grants access to the distance between the sensor and the reflecting object.

Its relative velocity is accessed through the frequency shift between the two signals, a.k.a. the Doppler effect. Indeed, the phase shift varies when the target is moving:

$$f_d = \frac{1}{2\pi}\frac{d\phi}{dt} = \frac{2v_R}{c} f_s, \tag{2.5}$$

where $v_R = dr/dt$ is the radial velocity of the target object w.r.t. the RADAR. This yields the frequency Doppler effect whereby frequency change rate between transmitted and received signals, $\frac{f_d}{f_s} = \frac{2v_R}{c}$, depends linearly on the relative speed of the reflector. Measuring this Doppler effect hence amounts to recovering the radial speed:

$$v_R = \frac{c f_d}{2 f_s}. \tag{2.6}$$

Radar's capability to discriminate between two targets with same range and velocity but different angles is called its angular resolution. It is directly proportional to the antenna aperture, that is, the distance between the first and last receiving antennas. The time delay between the received signals of each Rx transmitted by a given Tx carries the orientation information of the object.

The MIMO approach [Donnet & Longstaff 2006] is commonly used to improve the angular resolution without increasing the physical aperture: angular resolution increases by a factor of 2 for each added emitting antenna (Tx). Denoting N_{Tx} and N_{Rx} the number of its Tx and Rx channels respectively, a MIMO system builds a virtual array of $N_{\text{Tx}} \cdot N_{\text{Rx}}$ antennas. The downside of this approach is that the reflected signal of an object appears N_{Tx} times, making the data interleaved. The RADAR representation is easily de-interleaved considering a Low-Definition (LD) RADAR since it has at most 2 Tx antennas. This processing step applied to a HD RADAR is a greedy task; additional details are provided in Chapter 6.

Depending on the positioning of the antennas, the azimuth angle and the elevation of the object are respectively deduced from the horizontal and vertical pairs of (Tx, Rx). The Angle-of-Arrival (AoA) is deduced from the variation between the phase shift of adjacent pairs of Rx. In particular, the azimuth angle[1] α is obtained with the phase shift variation of horizontal adjacent Rx noted $\Delta\phi_\alpha = 2\pi f_s \frac{2h \sin\alpha}{c}$, where h is the distance separating the adjacent receivers.

[1] A geometric illustration is depicted in Figure 4.8

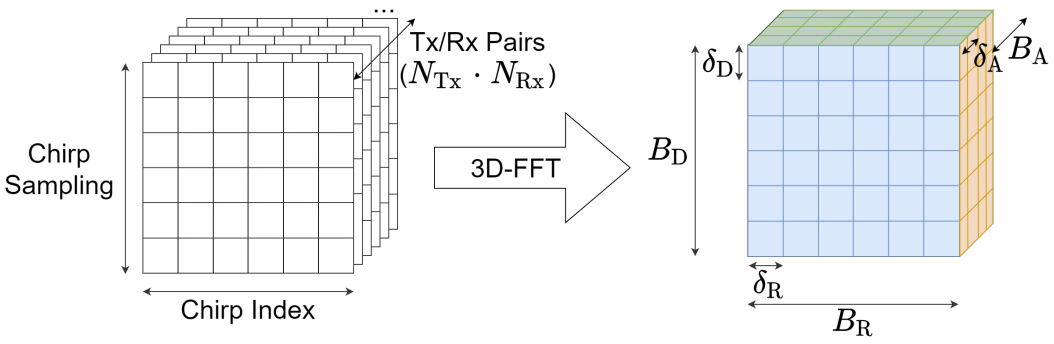


Figure 2.2: **Range-Angle-Doppler (RAD) tensor generation**. The signals received by all pairs of (Tx, Rx) antennas are transformed and stored in a 3D tensor in the frequency domain. Inverse FFT operations are respectively applied on the chirp index axis to deduce the range, on the chirp sampling axis (*i.e.* the frequency sampling for each chirp) for the Doppler and on the (Tx, Rx) pairs axis for the angle. The generated RAD tensor in the temporal domain has dimensions $B_{\text{R}} \times B_{\text{A}} \times B_{\text{D}}$. Each of its axes has a resolution of δ_{R}, δ_{A} and δ_{D} respectively depending on the RADAR sensor specificities. The geometry and the number of RADAR antennas define the recording resolutions in azimuth (δ_α) and elevation (δ_{E}) angles. Considering low-definition RADAR, the positioning of the antennas is only able to obtain the azimuth angle and $\delta_{\text{A}} = \delta_\alpha$.

2.2 Recordings and signal processing

2.2.1 Transformations in the temporal and frequency domains

A RADAR sensor transmits and receives signals, respectively noted s_E and s_R are written:

$$s_E(t) = A_E e^{j2\pi f_s t} = A_E e^{j\phi_E(t)}, \tag{2.7}$$

$$s_R(t) = A_R e^{j2\pi f_s (t-\tau)} = A_R e^{j\phi_R(t)}, \tag{2.8}$$

where A_E and A_R denote their amplitude. The phase shift induced by the time delay between the transmitted and received signal is computed by correlating the two signals. The convolution between two signals in the temporal domain is equivalent to a point-wise product of their Discrete Fourier Transform (DFT). Therefore, the transmitted and received signals are compared in the frequency domain.

The Fast Fourier Transform (FFT) algorithm applies a DFT to the data from the temporal domain to the frequency domain. Let $x : \{0, \cdots, N_T - 1\} \rightarrow \mathbb{C}$ be a sequence of N_T complex signals. For $\nu \in \{0, \cdots, N_f - 1\}$ a finite sequence of N_f frequencies, the DFT of x is defined as:

$$\hat{x}(\nu) = \sum_{t=0}^{N_T-1} x(t) e^{-j2\pi\nu\frac{t}{N_T}}. \tag{2.9}$$

A FFT is applied on N_T complex signals for each transmitted and received signals in the

temporal domain, their DFT is written:

$$\hat{s}_E(\nu) = \sum_{t=0}^{N_T-1} s_E(t) e^{-j2\pi\nu\frac{t}{N_T}}, \tag{2.10}$$

$$\hat{s}_R(\nu) = \sum_{t=0}^{N_T-1} s_R(t) e^{-j2\pi\nu\frac{t}{N_T}}. \tag{2.11}$$

The signals are compared in the frequency domain with a mixer that generates the Intermediate Frequency (IF) signal:

$$\hat{m}(\nu) = \hat{s}_E(\nu) \cdot \hat{s}_R(\nu). \tag{2.12}$$

The IF signal, illustrated in Figure 2.1, is deduced from the signal transmitted by Tx and received by Rx. The mixed signal is filtered using a low-pass filter and digitized by an analog-to-digital converter (ADC). This way, the recorded signal carries the Doppler frequencies, ranges and angles of all reflectors.

Consecutive filtered IF signals are stored in a frame buffer which is a frequency-domain 3D tensor: the first dimension corresponds to the chirp index; the second one is the chirp sampling defined by the linearly modulated frequency range; the third tensor dimension indexes (Tx, Rx) antenna pairs.

The recorded data are finally transformed in the temporal domain as illustrated in Figure 2.2. *E.g.* the inverse FFT applied on the chirp index axis processes an inverse DFT with N_f frequencies on the IF signals (Equation 2.12) as:

$$m(t) = \sum_{\nu=0}^{N_f-1} \hat{m}(\nu) e^{j2\pi t\frac{\nu}{N_f}}. \tag{2.13}$$

This method estimates the peak reached in the IF signal in the temporal domain due to time delay between the transmission and reception of the signals [Ghaleb 2009, Molchanov 2014]. Considering the chirp index, the distance of the reflector is then deduced from the estimated time delay.

The 3D tensor in the frequency domain (see Figure 2.2) is processed using an inverse three dimensional fast Fourier transform (3D-FFT): a range-FFT along the rows deducing the object range, a Doppler-FFT along the columns deducing the objects' relative velocity and an angle-FFT along the depth deducing the angle between two objects (see the range-FFT example in Equation 2.13). This sequence of inverse FFTs results in the Range-Angle-Doppler (RAD) tensor of dimensions $B_{\mathrm{R}} \times B_{\mathrm{A}} \times B_{\mathrm{D}}$, a 3D data cube of complex numbers where each axis amounts to discretised values of the corresponding physical measurement as illustrated in Figure 2.2.

The range, velocity and angle bins in the output tensor correspond to discretized values defined by the resolution of the RADAR. The range resolution is defined as $\delta R = \frac{c}{2B}$. The relative velocity, or Doppler resolution $\delta D = \frac{c}{2f_c T}$ is inversely proportional to the frame duration time. The angle resolution $\delta\alpha = \frac{c}{f_c N_{\mathrm{Rx}} h \cos(\alpha)}$ is the minimum angle separation

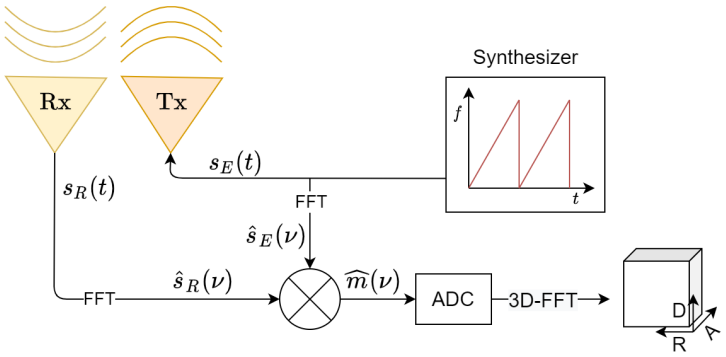


Figure 2.3: **Outline of the RADAR pipeline**. An electromagnetic wave is synthetized and transmitted in the environment with a Tx antenna. The received signal is mixed with the original one and stored in a tensor in the frequency domain. An inverse Fast Fourier Transform is applied on each axis of the recorded data to generate the Range-Angle-Doppler tensor in the temporal domain.

between two objects to be distinguished [Iovescu & Rao 2017], with N_{Rx} the number of Rx antennas and α the horizontal azimuth angle between the RADAR and an object reflecting the signal.

The outline of the RADAR pipeline, which is from the synthesis of the electromagnetic wave to the generation of the RAD tensor, is depicted in Figure 2.3. The following section will describe the speckle noise strongly represented in RADAR data.

2.2.2 Speckle noise

The intensities (squared modulus) in the RAD tensor provide information on the power backscattered by surrounding objects. A visualisation of the tensor by 2D pairs of axes is proposed in Appendix B.1.1. It can be noticed that information is redundant regarding each slice of the tensor considering its pairs of axes.

Moreover, the backscattered signal presents strong fluctuations also known as speckle phenomenon and which have been well-studied by Goodman [Goodman 1976]. The fluctuations are due to the addition of complex backscattered signals of elementary scatterers in the same resolution cell leading to a random intensity record in the absence of a significant reflector. The model proposed by Goodman assumes that a large number of echoes are produced by individual scatterers in a resolution cell with independent and identically distributed (iid) complex amplitudes. This phenomenon can be modeled as a multiplicative noise and can be written:

$$w = u \times s, \tag{2.14}$$

where w is the measured intensity in a resolution cell, u the underlying reflectivity and s the speckle [Dalsasso *et al.* 2021]. According to the work of [Goodman 1976] and considering multi-looked observations[2], the speckle can be modeled as a random variable following a

[2]The multi-looking method, used with Synthetic Aperture RADAR (SAR), consists in aggregating intensity

gamma distribution defined as:

$$p(s) = \frac{L^L}{\Gamma(L)} s^{L-1} e^{-Ls}, \tag{2.15}$$

where $L \geq 1$ is the number of looks and $\Gamma(.)$ the gamma function. As described by [Goodman 1976], $\mathbb{E}[s] = 1$ and $\mathbb{V}[s] = \frac{1}{L}$, the higher the number of looks, the lower the variance of the speckle.

The speckle becomes additive after a logarithmic transform applied to the recorded intensity while stabilizing its variance [Deledalle *et al.* 2017]. The log-transformed intensity $\tilde{w}$ is thus written:

$$\tilde{w} = \tilde{u} + \tilde{s}, \tag{2.16}$$

where $\tilde{s}$ is the log-speckle following a Fisher-Tippett distribution given by:

$$p(\tilde{s}) = \frac{L^L}{\Gamma(L)} e^{L\tilde{s}} e^{-Le^{\tilde{s}}}. \tag{2.17}$$

The expectation and variance of the log-speckle no longer depend on the underlined reflectivity. They are respectively noted $\mathbb{E}[\tilde{s}] = \psi(L) - \log(L)$ and $\mathbb{V}[s] = \psi(1, L)$, where $\psi(.)$ is the diagamma function and $\psi(.,.)$ is the polygamma function of order L [Abramowitz & Stegun 1965].

The multi-looking operation, consisting in averaging a few samples, reduces the noise strength, whereas the logarithmic transform induces a variance stabilization of the resulting signal [Deledalle *et al.* 2017, Dalsasso *et al.* 2021]. The next section will describe the transformations applied to the previously presented RAD tensor to obtain interpretable RADAR representations while reducing the speckle noise.

2.2.3 RADAR representations

Let $\mathbf{X}^{\text{RAD}}$ be the complex RAD tensor. Aggregating its intensities across one of its dimensions leads to three possible 2D views of the tensor: Range-Angle (RA), Range-Doppler (RD) and Angle-Doppler (AD) illustrated in Figure 2.4. For instance, we define the RA view, expressed in decibels, as:

$$\mathbf{x}^{\text{RA}}[r, a] = \Phi_{\text{D}}(\mathbf{X}^{\text{RAD}}), \tag{2.18}$$

where Φ_{D} is an arbitrary aggregation function compressing the RAD tensor through its Doppler dimension. As detailed in Section 2.2.2, an averaging followed by logarithm transform are commonly used in practice. In this work, we average the intensity of the RAD tensor on its third axis to reduce the speckle noise. Note that the hypothesis of independent and identically distributed samples is no longer verified along the third axis, which will have the consequence of attenuating the reflections[3]. With this method, the RA view

values of different looks which can be obtained in different ways.

[3]This limitation is discussed in Section 5.1.5 and Chapter 7.

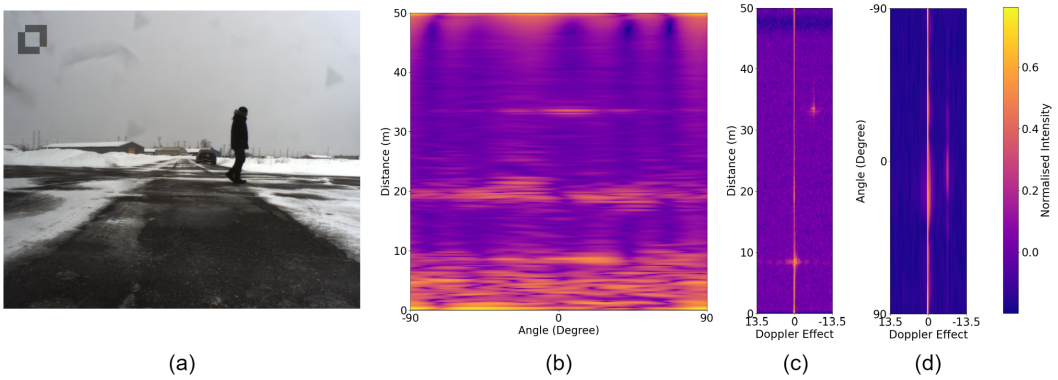


Figure 2.4: **An example of RADAR representations**. The Range-Angle-Doppler (RAD) tensor corresponding to the scene, illustrated with the camera image (a), is aggregated using Equation 2.19 to create the (b) Range-Angle, (c) Range-Doppler and (c) Angle-Doppler views corresponding to each 2D pair of axis of the tensor. This scene is a sample of the CARRADA dataset presented in Section 4.3.

is then defined as:

$$\mathbf{x}^{\mathrm{RA}}[r,a] = 10\log_{10}\left(\frac{1}{B_{\mathrm{D}}}\sum_{d=1}^{B_{\mathrm{D}}}\left|\mathbf{X}^{\mathrm{RAD}}[r,a,d]\right|^2\right), \tag{2.19}$$

with $|.|$ the modulus and B_{D} the number of Doppler bins. Turning RAD tensors into views aims both to compress substantially the data representation and to reduce its noise while preserving the objects' signature. As a consequence, the view representations are interpretable, offering a viable way to exploit the temporal dimension in tensor sequences. They also meet practical computational and memory constraints imposed by in-car embedding. Considering an example of application, it reduces the size of the data by a factor of 50[4]. An example of the RADAR views is illustrated in Figure 2.4.

With conventional FMCW RADARs, the RAD tensor is usually not available as it is too computationally intensive to estimate. A Constant False Alarm Rate (CFAR) algorithm [Rohling 1983] is typically applied to filter RD views. It keeps the highest intensity values while taking into account the local relation between points. The AoA is then computed for each high reflection to obtain a sparse RADAR point cloud, also called Direction-of-Arrival (DoA) point cloud, in Cartesian coordinates while each point has its Doppler and Radar Cross Section (RCS). To translate the AoA into an effective angle, one needs to calibrate the sensor. An alternative to the third FFT is to correlate in the complex domain the RD spectrum with a calibration matrix, to estimate the angles (azimuth and elevation). The complexity of this operation for a single point of the RD tensor is $\mathcal{O}(N_{\mathrm{Tx}}N_{\mathrm{Rx}}B_{\mathrm{A}}B_{\mathrm{E}})$, where B_{A} and B_{E} are respectively the number of bins over which azimuth and elevation

[4] As an example, the CARRADA dataset [Ouaknine *et al.* 2020], detailed in Section 4.3.1, provides RAD tensors of 17MB each. Their corresponding aggregated views are 0.1MB for RD/AD and 0.5MB for RA.

angles are discretized in the calibration matrix. A LD RADAR considers a single elevation ($B_{\mathrm{E}} = 1$) and the azimuth resolution is low. This process is therefore computationally affordable to generate sparse RADAR point clouds. However it meets a bottleneck with HD RADARs.

Multiple representations of RADAR data are available depending on the signal processing pipeline applied on the recordings: Analog-to-Digital Converter (ADC) data, RAD tensor, RD/RA/AD views or sparse point clouds. The fewer the pre-processing of the data, the more precise the signal corresponding to the objects is, the more cumbersome the representation is and the higher the level of noise. The point cloud is the lighter representation but the CFAR filtering has drawbacks: it drastically shrinks the shape of the objects' signature and loses small, low-reflection signatures, especially non-metallic objects such as pedestrians. Selecting a RADAR data representation depends on the application (classification, detection, semantic segmentation) and on a trade-off between volume of the data and the applied method. The following section introduces the background on ANN.

2.3 Artificial neural networks

2.3.1 Introduction

Machine learning is the process optimizing parameters of a statistical model to solve a task [Hastie *et al.* 2001]. The parameters of the model are adjusted to separate or combine the input data samples regarding redundant patterns. The supervised learning scheme assumes that each data sample has a label corresponding to the task to solve (*e.g.* categorizing the image of a cat, detecting cancer cells in medical images, predicting the values of a financial time series and so on). The optimization process consists in minimizing the error between the prediction of the model and the corresponding label. In other words, a machine learning algorithm is able to solve a task by learning to transform data.

Datasets are diverse and many different tasks can be learnt by a machine learning algorithm. The most common are classification (*e.g.* categorizing an image), detection (*e.g.* localizing an object in an image), regression (*e.g.* predicting the future price of a product), denoising (*e.g.* cleaning a corrupted sample) and machine translation (*e.g.* translating a sentence from a language to another).

Let consider a supervised learning setting, where $\mathbf{X} = \{\mathbf{x}_0, \cdots, \mathbf{x}_{N-1}\}$ is a set of data with N samples and $\mathbf{Y} = \{\mathbf{y}_0, \cdots, \mathbf{y}_{N-1}\}$ their corresponding ground-truth labels. A machine learning algorithm is a function noted $f_\theta(.)$ parameterized by a vector $\theta = (\theta_0, \cdots, \theta_{d-1})$, where d is the number of parameters in the model. It tries to predict a label for each sample of the dataset as

$$f_\theta(\mathbf{x}_i) = \hat{\mathbf{y}}_i, \tag{2.20}$$

where $\mathbf{x}_i \in \mathbf{X}$ and $\hat{\mathbf{y}}_i \in \hat{\mathbf{Y}} = \{\hat{\mathbf{y}}_0, \cdots, \hat{\mathbf{y}}_{N-1}\}$ is the prediction of the model.

In practice, the dataset is randomly divided into a training set and a testing set, respectively $\{\mathbf{X}^{\text{Train}}, \mathbf{Y}^{\text{Train}}\}$ and $\{\mathbf{X}^{\text{Test}}, \mathbf{Y}^{\text{Test}}\}$. It aims to estimate iid training and testing sets with a random separation of the entire dataset. Therefore, the algorithm will be evaluated on data that it has never seen during its training period.

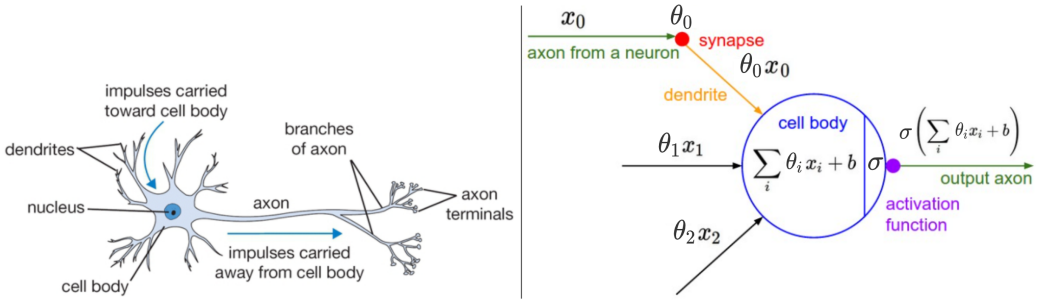


Figure 2.5: **Illustration of a biological neuron (left) and its mathematical model (right)**. [Karpathy 2021].

Let $\mathcal{L}(.,.)$ be a function quantifying the error between the ground truth and the predictions of the mode. The supervised learning optimization process tries to find a set of optimal parameters θ^* minimizing the error between the ground-truth labels and the predictions. During training, it is sought as:

$$\theta^* \in \arg\min_{\theta} \frac{1}{N^{\text{Train}}} \sum_{i=0}^{N^{\text{Train}}-1} \mathcal{L}(\mathbf{y}_i^{\text{Train}}, f_\theta(\mathbf{x}_i^{\text{Train}})), \tag{2.21}$$

where N^{Train} is the number of elements in $\mathbf{X}^{\text{Train}}$. Once the optimization process has converged, *i.e.* the training loss oscillates under a threshold, the generalization error of the model is evaluated as $\frac{1}{N^{\text{Test}}} \sum_{i=0}^{N^{\text{Test}}-1} \mathcal{L}(\mathbf{y}_i^{\text{Test}}, f_{\theta^*}(\mathbf{x}_i^{\text{Test}}))$, where $\frac{1}{N^{\text{Test}}}$ is the number of element in $\mathbf{X}^{\text{Test}}$, or with a defined evaluation metric related to the task to solve.

2.3.2 At the neuron level

In the human brain, a neuron receives electrical signals via *dendrites* which are processed in the *soma* (the cell body) and transmitted by the *axons* if the signal is above a threshold. The *synapses* finally transfer the signals to another neuron using the *axon* terminals. A representation is depicted in Figure 2.5 (left). A simple mathematical formulation of a biological neuron as illustrated in Figure 2.5 (right), we will name it Artificial Neuron (AN).

The Perceptron [Rosenblatt 1958] algorithm is the simplest AN optimizing its weights via a machine learning process. It is a supervised linear classifier predicting a binary outputs as

$$f_\theta(\mathbf{x}) = \begin{cases} 1 & \text{if} \quad \theta \cdot \mathbf{x} + b > 0, \\ 0 & \text{otherwise}, \end{cases} \tag{2.22}$$

where b a bias scalar value. The optimization process will converge only if the data are linearly separable, which is usually not the case. An activation function, noted $\sigma(.)$, is introduced to mimic the *axon* behavior thresholding the signals in a biological neuron. This function aims to introduce a non-linearity and extend its capacity to separate the data.

The mathematical formulation of an AN is written

$$f_\theta(\mathbf{x}) = \sigma(\theta \cdot \mathbf{x} + b), \tag{2.23}$$

which is equivalent to the Perceptron when σ is a Heaviside step function. The most common activation functions are:

- The sigmoid function "squashing" the values into a range of $[0, 1]$: $\sigma(a) = \frac{1}{1+\exp(a)}$;
- The hyperbolic tangent (tanh) "squashing" the values into a range of $[-1, 1]$: $\sigma(a) = \frac{2}{1+\exp(-2a)} - 1$;
- The Rectified Linear Unit (ReLU) function thresholding the value to zero: $\sigma(a) = \max(0, a)$;
- The Leaky Rectified Linear Unit (LeakyReLU) function allowing a small negative slope of the input value: $\sigma(a) = \max(\varepsilon a, a)$, where ε is the slope.

Note that the activation functions have linear, or approximately linear, regions according to their limits. An activation is saturated when it reaches a linear region and has a constant, or almost constant value.

When the sigmoid function is used as the activation function, the neuron behaves like a binary linear classifier, a.k.a. logistic regression, providing a probability to belong to a class. The activation function is chosen according to the problem to solve and the expected distribution of the data.

2.3.3 At the layer level

The Perceptron algorithm is not able to solve exactly the XOR problem which is not linearly separable [Minsky & Papert 1969]. The MLP algorithm leverages this limitation by considering multiple Perceptrons stacked together to form a layer. Multiple layers of neurons are considered in the MLP to learn specific patterns and non-linear dependencies between the input data as illustrated in Figure 2.6. The MLP is a special case of the ANN where each neuron of its hidden layer is connected to each neuron of the next layer with a unique weight. It is also named fully connected layer. Note that there is no connection between the neurons of the same layer. The MLP algorithm consists in three types of layers:

- The input layer receiving the data as input of the model;
- The hidden layer which is a fully connected layer transforming and transmitting the input signal to the next layer. Note that several hidden layers can follow each others;
- The output layer returning the predictions of the model with one or multiple neurons depending on the task to solve (*i.e.* binary classification, multi-class classification, multi-value regression and so on).

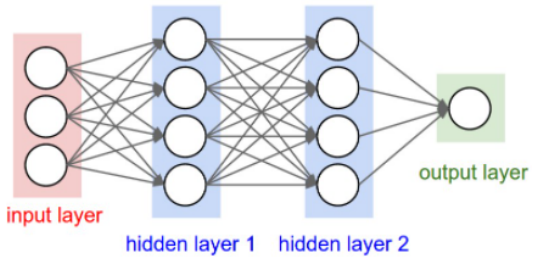


Figure 2.6: **Illustration of a Multilayer Perceptron**. The illustrated algorithm [Karpathy 2021] is a 3-layer neural network with three inputs, two hidden layers of four neurons each and a single output layer. There is no connection between the neurons of the same layer.

2.3.4 Training a neural network

Training an ANN means adapting the weights associated to each neuron in order to minimize a loss function. The optimization process iteratively updates the weights of the network by quantifying their gradient w.r.t. the loss function. This function must be differentiable and has to be chosen carefully according to the problem to solve.

Loss functions Considering a regression task, the Mean Squared Error (MSE) loss function is commonly used to minimize the gap between the true and the predicted values. The objective is formalized as

$$\min_{\theta} \frac{1}{N} \sum_{i=0}^{N-1} (\mathbf{y}_i - f_\theta(\mathbf{x}_i))^2, \tag{2.24}$$

where N is the number of elements in $\mathbf{X}$.

The binary classification task is tackled using a Binary Cross-Entropy (BCE) loss function. It quantifies the likelihood to predict the true labels. In practice and under the assumption of an activation function defined in $[0, 1]$, the optimization process minimizes the negative log-likelihood;

$$\min_{\theta} -\frac{1}{N} \sum_{i=0}^{N-1} \mathbf{y}_i \log(f_\theta(\mathbf{x}_i)) + (1 - \mathbf{y}_i) \log(1 - f_\theta(\mathbf{x}_i)). \tag{2.25}$$

The multi-class classification consists in predicting for each K class a probability that an input $\mathbf{x}_i$ belongs to each one of them. The ground-truth $\mathbf{y}_i$ is a one-hot vector of K elements, with 1 at the corresponding index of the class, 0 otherwise. The general optimization problem for multi-class classification using a categorial Cross-Entropy (CE) is written:

$$\min_{\theta} -\frac{1}{K \cdot N} \sum_{k=0}^{K-1} \sum_{i=0}^{N-1} \mathbf{y}_{k,i} \log(f_\theta(\mathbf{x}_i)_k), \tag{2.26}$$

where $\sum_{k=0}^{K-1} \log(f_\theta(\mathbf{x}_i)_k) = 1$.

Gradient descent optimization The weights of a neural network are initialized before starting the optimization process. The initialized set of weights $\theta \in \mathbb{R}^d$ denotes the starting point of the process in a d dimensional space. Initialization is an important choice as it leads to different local minima, and thus to performance variability.

Considering a neural network with a randomly initialized vector of weights, their values are usually randomly drawn from a Gaussian or uniform distribution to avoid extreme values in long tail distributions. There is no better choice between these two distributions [Goodfellow *et al.* 2016]. However, the initialisation has a significant impact on both the optimization process and the generalisation capacities of the network. Practical examples of uniform distributions used for the weight initialisation are $\mathcal{U}(-0.3, 0.3)$, $\mathcal{U}(0, 1)$ or $\mathcal{U}(-1, 1)$.

The Xavier initialization method [Glorot & Bengio 2010] is commonly used and consists in randomly drawing the initial weight values in $\mathcal{U}(-\frac{1}{\sqrt{n}}, \frac{1}{\sqrt{n}})$, where n is the number of inputs of the initialized layer. An extension named normalized Xavier initialisation considers the uniform distribution defined as $\mathcal{U}(-\sqrt{\frac{6}{n+m}}, \sqrt{\frac{6}{n+m}})$, where m is the number of neurons in the initialized layer. This initialization method is generally used to train neural network architectures because the magnitude of the values is well defined w.r.t. the input and current layers.

A common practice in deep learning, called pre-training or transfer learning, consists in initializing the model with a weight vector that has already been learnt on a pretext task with a large annotated dataset. The pre-trained parameters are then fine-tuned on a down-stream task, usually with fewer annotations available, benefiting from the knowledge acquired in the previous optimisation process.

The optimization process considers an arbitrary differentiable loss function $\mathcal{L}(.,.)$. The error between the ground truth labels and the predictions of the model $f_\theta(.)$ is quantified as $\frac{1}{N}\sum_{i=0}^{N-1} \mathcal{L}(\mathbf{y}_i, f_\theta(\mathbf{x}_i))$ as described in the previous paragraphs. The optimization method commonly used during the training process is the Stochastic Gradient Descent (SGD) [Robbins & Monro 1951, Kiefer & Wolfowitz 1952]. It updates the weights iteratively in a direction opposite to the gradient of the loss function. Computing the gradients w.r.t. the loss is not scalable considering the entire training dataset. Let's consider a "batch" of B samples, written $\mathbf{X}_B = \{\mathbf{x}_0, \cdots, \mathbf{x}_{B-1}\}$, the SGD method is defined as:

$$\theta_t := \theta_{t-1} - \frac{\eta}{B}\nabla_\theta \sum_{i=0}^{B-1} \mathcal{L}(\mathbf{y}_i, f_\theta(\mathbf{x}_i)), \tag{2.27}$$

where $\eta > 0$ is the learning rate, θ_t the parameters of the model at iteration t and $\mathbf{x}_i, \mathbf{y}_i$ the i-th sample of the batch with its associated label.

The learning rate is a key hyperparameter defining the amplitude of the weight update. It is chosen carefully because if it is small, the optimization will reach a local minima with numerous number of steps. And if it is high, the process may diverge.

The optimization landscape of a neural network training algorithm is complex and non-

convex. Thus it converges to a local minima which is difficult to characterize. Exploring gradient descent algorithms and the optimization landscape of neural networks is an active field of research. In the presented experiments, the ADAM [Kingma 2015] method is used for optimization. It is an efficient process considering momentum of the gradients to remember their past trajectories. The parameter update depends on m_t the estimation of the bias-corrected first moment and v_t the estimation of the bias-corrected second moment. Considering a batch of B samples, the iterative gradient descent method proposed by [Kingma 2015] is defined as:

$$m_t := \frac{\beta_1 m_{t-1} + \frac{(1-\beta_1)}{B}\nabla_\theta \sum_{i=0}^{B-1} \mathcal{L}(\mathbf{y}_i, f_\theta(\mathbf{x}_i))}{(1-\beta_1^t)}, \tag{2.28}$$

$$v_t := \frac{\beta_2 v_{t-1} + \frac{(1-\beta_2)}{B}\nabla_\theta^2 \sum_{i=0}^{B-1} \mathcal{L}(\mathbf{y}_i, f_\theta(\mathbf{x}_i))}{(1-\beta_2^t)}, \tag{2.29}$$

$$\theta_t := \theta_{t-1} - \eta \frac{m_t}{\sqrt{v_t} + \varepsilon}, \tag{2.30}$$

where $\beta_1, \beta_2 \in [0, 1)$ the exponential decay rates of the momentum estimations, $\varepsilon = 10^{-8}$ and β_1^t, β_2^t denoting respectively β_1, β_2 to the power t.

Forward and backward propagations During the training process, a feedforward pass through the network is performed to infer the predictions regarding an input. The loss function is evaluated by comparing the predictions and the ground-truth labels. Then the back-propagation step aims to evaluate the gradients of the weights regarding the loss function thanks to the chain rule.

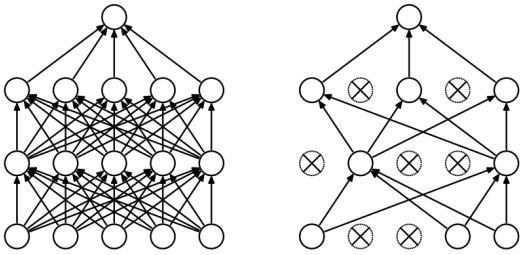
Let $\theta = (\theta_0, \cdots, \theta_{d-1})$ be the vector of d weights of a neural network. The network can be represented as an oriented graph where each neuron is a vertex. The neurons are connected to the previous and next layers via edges corresponding to the weights of the network.

According to the chain rule, the gradient of a weight will integrate all other weights on which it depends to calculate the loss, *i.e.* the path on the graph that leads to the loss. Considering that θ_n is the last edge leading to the loss computed at the last neuron, the gradient of θ_i w.r.t. the loss depends on $(\theta_i, \cdots, \theta_n)$. It is computed as:

$$\nabla_{\theta_i} \mathcal{L}(\mathbf{y}, f_\theta(\mathbf{x})) = \frac{\delta \mathcal{L}(\mathbf{y}, f_\theta(\mathbf{x}))}{\delta \theta_i} = \sum_j \frac{\delta \mathcal{L}(\mathbf{y}, f_\theta(\mathbf{x}))}{\delta f_{\theta_j}(\mathbf{x})} \frac{\delta f_{\theta_j}(\mathbf{x})}{\delta \theta_i}, \tag{2.31}$$

where $f_{\theta_j}(\mathbf{x})$ is the sub-function, *i.e.* the sub-graph, of $f_\theta(\mathbf{x})$ parameterized by $\{\theta_j, \cdots, \theta_n\}$ weights. We suggest that the reader refers to the work of [Goodfellow *et al.* 2016] for in-depth details on graph computing for back-propagation.

Regularization A machine learning model is over-parameterized if its number of parameters is much larger than the number of training samples. As a consequence, it has the capacity to learn the data itself, in other words to over-fit the training set, leading to a lack of generalization and poor performances on the test set. This problem is tackled using



(a) Standard Neural Net (b) After applying dropout.

Figure 2.7: **Example of Dropout**. Example of the Dropout method [Srivastava *et al.* 2014] with a probability of 0.5.

regularization methods to constraint the parameters of the model during the optimization process.

The L1-norm is useful to reduce the dimension of the parameter space by forcing their value toward zero. The optimization is written

$$\min_{\theta} \sum_{i=0}^{N-1} \mathcal{L}(\mathbf{y}_i, f_\theta(\mathbf{x}_i)) + \frac{\lambda}{2} \|\theta\|_1 , \tag{2.32}$$

where λ is an hyperparameter to scale the regularization strength. The L2-norm used as a regularization method aims to penalize extreme values of the weights. It reduces over-fitting on outlier data. The optimization is written

$$\min_{\theta} \sum_{i=0}^{N-1} \mathcal{L}(\mathbf{y}_i, f_\theta(\mathbf{x}_i)) + \frac{\lambda}{2} \|\theta\|_2^2 . \tag{2.33}$$

The dropout method [Hinton *et al.* 2012, Srivastava *et al.* 2014] regularizes neural networks by keeping a few active neurons with a propability p at each iteration of the training. The targeted neurons are randomly chosen at each optimization step. Figure 2.7 illustrates this method with $p = 0.5$. During the forward propagation, the neurons of the i-th layer transmit the signal $f^i_{\theta_i}(\mathbf{x})$ to the $(i+1)$-th layer. Let $\mathbf{z}_{i+1}$ be a binary vector taking 1 with probability p and 0 with probability $1-p$. The neurons at the layer $i+1$ are selected as:

$$f^{i+1}_{\theta_{i+1}}(\mathbf{x}) = \mathbf{z}_{i+1} \odot (f^i_{\theta_i}(\mathbf{x}) \cdot \theta_i + b_i). \tag{2.34}$$

At training time, each $\mathbf{z}_i$ is randomly drawn during the forward pass, they are fixed during the backward pass. A test time, the dropout method is not applied but a scaling factor of $\frac{1}{1-p}$ over the set of weights compensates the high number of neurons alive. The dropout method prevents over-fitting by reducing artificially the number of parameters to train, it is commonly used in fully connected layers.

2.4 Convolutional neural network

2.4.1 Convolutional layer

Natural images recorded from cameras are presented as 3D matrices. Their pixels are in the range $[0, 255]$ for each Red, Green and Blue (RGB) channels. Using images as the input of a machine learning model, in particular considering a fully connected ANN, is computationally expensive. As an example, considering an image of dimension $200 \times 200 \times 3$, a single fully connected layer will be represented by 120,000 parameters to process each pixel. The previous ANN cannot scale to high dimensional input representations. Moreover, it is not able to learn spatial features directly from the input data.

In recent years, CNNs using convolution operations has been explored to tackle these issues. Let $f \in \mathbb{R}^{H \times W \times C}$ an image or feature map with C channels and $g \in \mathbb{R}^{k_1 \times k_2 \times C}$ a set of C 2D kernels. The 2D convolution operator is formally written

$$(f * g)(h, w) = \sum_{i=-\infty}^{+\infty} \sum_{j=-\infty}^{+\infty} \sum_{c=-\infty}^{+\infty} f(i, j, c) g(h - i, w - j, c), \tag{2.35}$$

where $h, w \in \mathbb{N}$ defines the coordinates in the output feature map. The equivariance to translation property implies that the convolution of a shifted input contains the same information as the original one. As a consequence, the operation is able to detect patterns anywhere in the input.

A convolutional layer convolves a 2D kernel, over the entire input. At each iteration, it performs a dot product between the group of local values and the kernel, producing a scalar value. The scalar values are successively stored to create a feature map. This type of layer aims to learn local and spatial features, while reducing the number of parameters, making convergence easier than in the case of fully-connected layers. It is particularly well suited to high dimensional data, composed of redundant geometric and textural patterns, *e.g.* images. Moerover, the weights of a convolutional layer kernel are shared on the entire image meaning that the number of parameters regarding a fully connected ANN is dramatically reduced.

As the convolution operation is differentiable, its kernels of weights can be learnt using the back-propagation method [LeCun *et al.* 1989] with gradient descent regarding a loss function.

2.4.2 Complementary methods and layers

Practical methods have been introduced to better adapt to the dimensions of the image and to reduce the computational cost of the convolution operation. The “padding” method consists in adding default values at the border of the input to simple handle boundary conditions due to the convolution. Regarding the size of the kernel, additional values are required to create a feature map with the same size than the input. The default values are commonly set to zero (zero-padding) but many other methods are employed as duplicating the border values or replicating values as a mirror. The “stride” method aims to slide the convolutional kernel on the input with a pixel gap at each iteration similarly as a down-

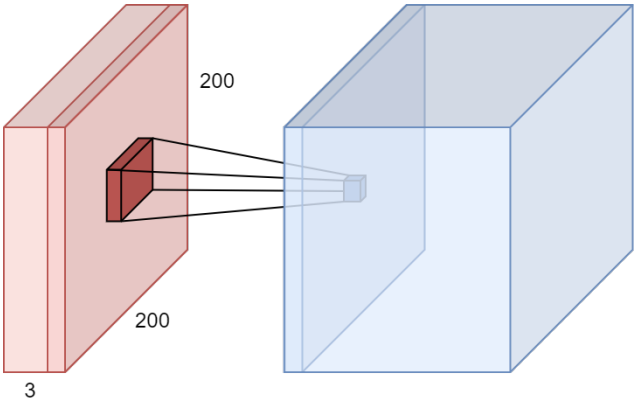


Figure 2.8: **Example of a convolutional layer**. An $200\times200\times3$ input volume (light red) is convolved by a kernel (dark red) producing a scalar value (small blue cube) at each convolution step. The kernel creates a 2D feature maps once it has sliced the entire input. The convolutional layer produces as many feature maps as kernels to learn; the output is a 3D tensor of feature maps.

sampling method. A kernel usually convolves an image with a stride of 1 pixel. This parameter can be increased to reduce the computational cost of the operation, *e.g.* a stride of 2 will slide the kernel with a gap of 1 pixel between each matrix product and thus, produces an output down-sampled by a factor of 2. However, small patterns may be missed. The stride is usually increased when a convolution is applied to high resolution images. Let D be the spatial dimension of input (either height or width), S the stride of the convolution, P the additional padding dimension and k the size of the kernel. The dimension of the output after applying the convolution operation is $\frac{D-k-2S}{P}+1$.

The architecture of a neural network dictates the composition of its successive layers. A CNN extracts information from the input data with successive convolutional layers [Krizhevsky *et al.* 2012]. Each one of these layers takes as input the output feature maps of the previous layer. To reduce the computational cost of such models and learn features at different scales, the size of the representations is progressively reduced. A pooling layer is nested between two convolutions applying a $\max(.)$ operation on the intermediate feature map as illustrated in Figure 2.9. As the convolution, it slides the input with a kernel (usually 2×2) and stores the maximum value in the outputted feature map. The padding and stride methods are also applicable in the max-pooling layer.

Recent advances in the understanding of neural networks optimization schemes lead to a better control on the gradient descent algorithms used during training. The distribution of the inputs of each layer changes during training due to the initialization of the weights, the intrinsic randomness of the data and the changes in the distribution of the weights in the previous layer. This phenomena, named internal covariate shift [Ioffe & Szegedy 2015], slows down the training (requires low learning rates) and may lead to divergence due to saturated non-linearities. The batch normalization layer introduced by [Ioffe & Szegedy 2015] refining the distribution of the inputs of each layer during training to reach a Gaussian dis-

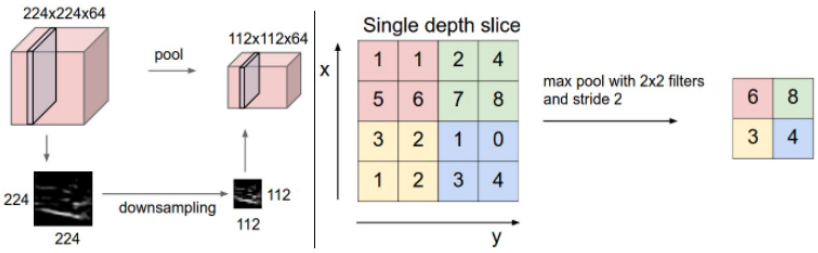


Figure 2.9: **Example of a max-pooling layer**. Left: example of the application of a pooling layer on a 200×200×32 input with a kernel of size 2×2 and a stride of 2. Right: detail of the application on local regions of the input [Karpathy 2021].

tribution $\mathcal{N}(0,1)$. The batch normalization layer has a significant impact on convolutional networks by smoothing the optimization landscape to propagate relevant information with the gradients [Goodfellow *et al.* 2016]. Let $\mathbf{x}$ be the input of a layer over a mini-batch $\mathcal{B} = \{\mathbf{x}_0, \cdots, \mathbf{x}_{m-1}\}$. The layer first normalizes $\mathbf{x}$ w.r.t. to the statistics of the mini-batch as

$$\mu_{\mathcal{B}} = \frac{1}{m}\sum_{i=0}^{m-1}\mathbf{x}_i, \tag{2.36}$$

$$\sigma^2_{\mathcal{B}} = \frac{1}{m}\sum_{i=0}^{m-1}(\mathbf{x}_i - \mu_{\mathcal{B}})^2, \tag{2.37}$$

$$\hat{\mathbf{x}}_i = \frac{\mathbf{x}_i - \mu_{\mathcal{B}}}{\sqrt{\sigma^2_{\mathcal{B}} + \varepsilon}}, \tag{2.38}$$

where $\hat{\mathbf{x}}_i$ is the normalized sample of the mini-batch input and $\varepsilon > 0$ a negligible value. The normalized sample is then scaled and shifted as

$$\mathbf{o}_i = \gamma\hat{\mathbf{x}}_i + \beta, \tag{2.39}$$

where $\mathbf{o}_i$ is the output sample of the batch normalization layer and γ, β two learnt parameters. This layer is commonly applied after a convolutional layer to normalize the distribution of the transformed data. It is usually followed by an activation layer avoiding the saturation of the non-linearity. The following section will briefly introduce recurrent neural network structures.

2.5 Recurrent neural network

A RNN is specialized in processing sequence of data. It is particularly used in temporal series analysis (*e.g.* for music, video or stock market) and Natural Language Processing (NLP) (*e.g.* textual analysis or translation). Its structure is based on a cell taking as input an element of the sequence and an activation vector. It outputs an hidden state and an

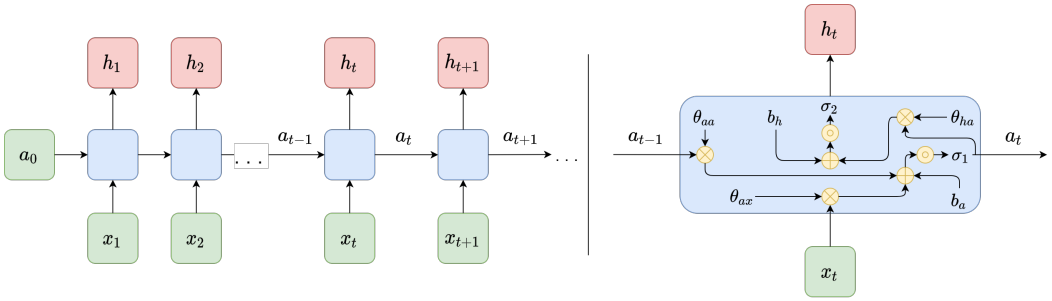


Figure 2.10: **Example of a recurrent neural network**. A traditional recurrent neural network processes a sequence iteratively and produces an hidden state (h_t) and an activation vector (a_t) (left). The simplest cell consists in applying three fully connected layers at different level of the data processing (right).

activation vector. At each iteration, the cell of the network processes the next element of a sequence with the activation vector produced at the previous step as illustrated in Figure 2.10 (left). Considering an application where a sequence is used to predict another sequence, the hidden states produced at each step correspond to the predictions of the network.

This type of network can be formalized differently depending on the task to solve. The RNN presented and illustrated in 2.10 (left) is a specific case of a many-to-many application because it takes a sequence as input and outputs a sequence, both having the same length. It is generally used for named entity recognition. A many-to-many application with different length of sequences is used for text translation, the input sequence of words is processed completely and the predicted sequence, in another language, is generated afterward. The many-to-one processes an entire sequence to generate a single output, *e.g.* for sentiment classification in texts. Finally, the one-to-many consists in generating a sequence by considering a single observation as input.

In the many-to-many application presented in Figure 2.10, considering same length sequences, the cell of the RNN defines the operations applied to the element of the sequence and the activation vector. Let $\theta_{aa}, \theta_{ax}, \theta_{ha}, b_a, b_h$ be trainable coefficients shared temporally and σ_1, σ_2 activation functions. The traditional RNN cell, illustrated in Figure 2.10 (right), is defined as:

$$\mathbf{a}_t = \sigma_1(\theta_{aa}\mathbf{a}_{t-1} + \theta_{ax}\mathbf{x}_t + b_a), \tag{2.40}$$

$$\mathbf{h}_t = \sigma_2(\theta_{ha}\mathbf{a}_{t-1} + b_h), \tag{2.41}$$

where $\mathbf{x}_t$ is the element of the input sequence at time t, $\mathbf{a}_t$ and $\mathbf{h}^t$ are respectively the activation and hidden state vectors produced at time t.

The RNN models have multiple benefits: they process sequences of any length, the number of parameters does not increase with the input size, they consider the historical information and the parameters are shared at each timestamp. However, they are really

slow to train and suffer from vanishing gradients which are computed at the end of the sequence. Long-term dependencies are also difficult to learn because they are lost in the processing of sequences.

The Long Short-Term Memory (LSTM) cell [Hochreiter & Schmidhuber 1997] aims to alleviate these limitations with a particular structure learning long and short-term dependencies while easily back-propagating the gradients during time. The details of RNN cell structures are beyond the scope of this thesis. We suggest that the reader refers to the work of [Goodfellow *et al.* 2016] for in-depth information. Many best practices and modules have been developed in the conception of neural network architecture for the past few years. The next section aims to provide intuitions on the widely used methods in deep learning for scene understanding, in particular for classification, object detection and semantic segmentation.

2.6 Deep learning

Deep learning models are composed of multiple layers stacked one after the other forming an end-to-end differentiable model trained via back-propagation. Layers and neural network architectures have been widely explored in the past few years improving the feature representations of the data. Recent large scale annotated datasets helped to train models with increasing the number of parameters while improving the performances on public benchmarks and challenges.

This section will focus on deep learning applications for computer vision, in particular on well-known tasks in scene understanding namely classification, object detection and semantic segmentation. Examples of these applications are illustrated in Figure 2.11. This section aims to provide a background on existing layers and deep learning architectures which will be useful for the following chapters. The reported performances in Sections 2.6.1, 2.6.2 and 2.6.3 are evaluated on different tasks and datasets, thus they can not be directly compared *per se*.

2.6.1 Classification

One of the most popular task in computer vision is classification, *i.e.* associating a category to each image of a dataset. Classification has been widely explored using the ImageNet dataset [Deng *et al.* 2009]. A collaboration between Stanford University and Princeton University led to this large scale dataset of fourteen millions images annotated in one thousand categories. They created the annotations using set of synonym rings (or synsets)[5]) of the WordNet[6] lexicon tree. The original challenge consisted in a simple classification task, each image belonging to a single category among one thousand, from specific breed of dog to precise type of food. Due to its large scale, the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) is one of the most popular in computer vision. For clarity, we will call it the ImageNet challenge. This section will detail advances in deep learning architectures and methods which have continuously improved the performances on the ImageNet

[5] A synonym ring or synset, is a group of data elements that are considered semantically equivalent.
[6] https://wordnet.princeton.edu/

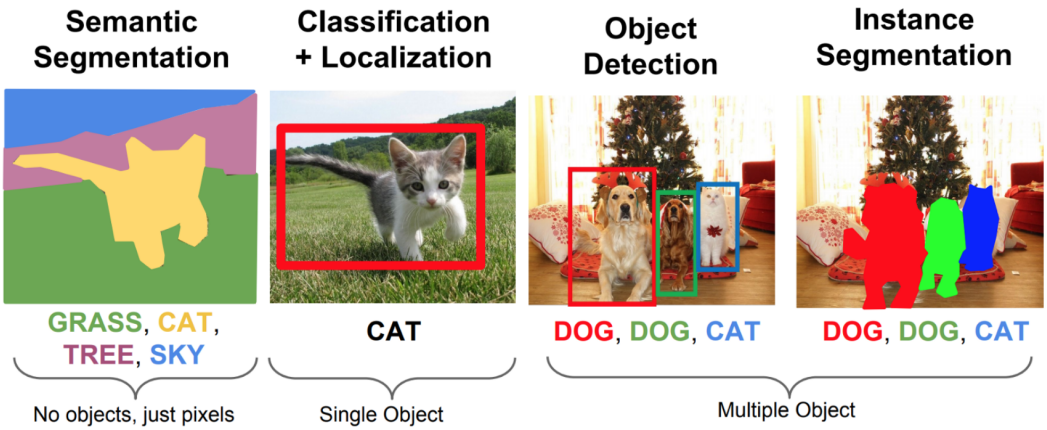


Figure 2.11: **Comparison between different computer vision applications**. Examples of well-known computer vision tasks for scene understanding with one of multiple objects to recognize.

challenges from 2012 to 2018. In the following paragraphs, we note the top-k error rate as the inverse of the top-k accuracy, also written $1 - \frac{\text{TP+TN}}{\text{TP+FP+TN+FN}}$, where a prediction is considered as positive if it is ranked in the top-k highest probabilities considering the predictions for all classes; and TP the true positive, TN the true negative, FP the false positive and TN the false negative.

The advent of deep learning (AlexNet). The ImageNet challenge has been traditionally tackled with image analysis algorithms such as SIFT [Lowe 2004] with mitigated results until the late 90's. However, a leap in performances has been brought by using neural networks. Inspired by [LeCun *et al.* 2012], the first deep learning model proposed by [Krizhevsky *et al.* 2012] drew attention to the public by beating all the previous computer vision methods with a top-5 error rate of 15.3%. The proposed AlexNet model can be considered today as a simple architecture with two consecutive convolutional, max-pool layers and three fully-connected layers.

Going deeper (VGG). In 2014, [Simonyan & Zisserman 2015] proposed the VGG16 architecture, composed of sixteen convolutional layers, four nested max-pool layers and three final fully connected layers. One of its specificities is to chain multiple convolutional layers with ReLU activation functions creating non-linear transformations. The authors also introduced kernels of weights of size 3×3 for each convolution (as opposed to 11×11 filters in the AlexNet model). They noticed that similar patterns can be learnt with smaller kernels while decreasing the number of parameters to learn. Using smaller kernels allows more convolutional layers to be stacked. As a consequence, deep layers have a larger receptive field. They thus have the capacity to learn fine-grained patterns at different scales. With these methods, the authors reduced by a factor of two the error rate of the AlexNet model reaching a top-5 error rate of 7.3% on the 2012 ImageNet challenge.

Inception modules (GoogLeNet and Inception V2). The "inception module" has been inspired by the work of [Lin *et al.* 2014a], consisting in training successive convolutional layers while introducing MLP between two layers. This idea have been exploited by [Szegedy *et al.* 2015] who proposed GoogLeNet (a.k.a. Inception V1), a deep neural network with 22 layers of inception modules for a total of over 50 convolutional layers. Their proposed module is composed of parallel convolutions with 1×1, 3×3, 5×5 kernels of weights and a 3×3 max-pool layer to increase the sparsity in the model. The produced feature maps are then concatenated and analyzed by the next inception module. The error rate on the 2012 ImageNet challenge has decreased to 6.7% while requiring significantly less memory than the VGG16 architecture (55MB *v.s.* 490MB). This gap is due to the three more fully-connected layers in the VGG.

In 2015, [Szegedy *et al.* 2016] developed the Inception V2 model, mostly inspired by the first version. The authors have changed the 5×5 kernel in the inception modules by two 3×3 kernels. It reduces the computational cost of the model which reached a top-5 error rate of 5.6% on the ImageNet challenge. The authors also proposed to factorize the convolution kernels of an inception module. It consists in replacing a convolution with a 3×3 kernel with two convolutions of kernels 3×1 and 1×3 respectively. This method reduces the number of parameters and the computational cost of the model. The Inception V3 architecture is composed of inception modules with factorized kernels. The authors also changed the first layers to process higher resolution inputs. They finally reached a top-5 error rate of 3.58% on the 2012 ImageNet challenge.

Residual learning (ResNet). In their work, [He *et al.* 2016] noticed that extremely deep models are difficult to train, leading to decreasing performances. They introduced the "residual learning" method creating a connection between the output of one or multiple convolutional layers and their original input with an identity mapping. In other words, the model tries to learn a residual function keeping most of the information.These connections also help to better propagate the gradients through deep networks by maintaining their magnitude. Residual learning neither requires any additional parameters, nor increases the computational complexity of the model. The authors proposed several architectures, named ResNet-X, where X is the number of convolutional layers with 3×3 kernels using residual learning by blocks of two layers. The ResNet-152 performed a top-5 error rate of 4.49 % on the 2012 ImageNet challenge (less that the inception V3) and it won the 2015 challenge with a top-5 error rate of 3.57%.

Residual learning and inception modules (Inception-ResNet). [Szegedy *et al.* 2017] have combined inception modules increasing the sparsity and residual blocks to learn deeper networks. The residual inception blocks are stacked with a similar architecture to the Inception V3 model. It results in the Inception V4[7], or Inception-ResNet, which can be trained faster and outperforms other methods with a top-5 error rate of 3.08% on the 2012 ImageNet challenge.

[7] [Szegedy *et al.* 2017] developed a pure (*i.e.* without residual block) Inception V4 and an Inception-ResNet V2 model which uses inception modules and residual blocks. The aforementioned Inception V4 is the Inception-ResNet V2 providing the best performances.

Squeeze and excitation module. The "Squeeze-and-Excitation" module has been introduced by [Hu *et al.* 2018]. It performs a global pooling on the input feature maps, followed by a fully connected layer with ReLU activation, and a second fully connected layer with Sigmoid activation. The output is scaled by to the input resolution and a residual connection is performed. Its main advantage is the low number of parameters due to the downsampling applied to the input maps. It won the 2017 ImageNet challenge with a top-5 error rate of 2.25%.

Conclusion This section described the milestones reached in deep learning for the image classification task, in particular applied to the ImageNet challenge. Well-known modules and architectures have be detailed and they are still commonly used for feature extraction nowadays. However, it is not an exhaustive list of all the existing models between 2012 and 2018. The advances from 2018 to date, in particular on Transformers architectures [Dosovitskiy *et al.* 2021, Liu *et al.* 2021d, Touvron *et al.* 2021], are beyond the scope of this thesis.

2.6.2 Object detection

Classification methods detailed in Section 2.6.1 categorize images into a single class, usually corresponding to the most salient object[8]. Images, *e.g.* recorded in urban scenes, are usually complex and contain multiple objects. In this case, such models are uncertain about which label to assign. The object detection task consists in localising and classifying all the objects in an input by predicting a bounding box around each one of them. It is therefore more appropriate for complex scene understanding. In this section, well-known datasets and evaluation metrics are briefly introduced. Then, each paragraph will detail a commonly used deep neural network architecture for object detection. This list is not exhaustive and additional methods are detailed in Appendix A.1.1.

Datasets. The PASCAL Visual Object Classification (PASCAL VOC)[9] dataset [Everingham *et al.* 2015] is a well-known dataset for classification, object detection and segmentation of objects. There are eight challenges spanning from 2005 to 2012, each of them having its own specificities. Considering the object detection task, there are around 10,000 images for training and validation with bounding boxes covering 20 categories.

Since 2013, ImageNet [Russakovsky *et al.* 2015] has released an object detection challenge with bounding boxes. The training dataset is composed of around 500,000 images only for training and 200 categories. It is rarely used because the size of the dataset requires a large computational power for training. Also, the high number of classes is difficult to tackle considering the object recognition task. A comparison between the 2014 ImageNet dataset and the 2012 PASCAL VOC dataset is available online [10].

The Common Objects in COntext (COCO)[11] dataset [Lin *et al.* 2014b], developed by

[8] Misleading examples in ImageNet have a single label although several objects are visible.
[9] http://host.robots.ox.ac.uk/pascal/VOC/
[10] http://image-net.org/challenges/LSVRC/2014/
[11] http://cocodataset.org

Microsoft, proposes four challenges: caption generation, object detection, key point detection and object segmentation. This section focuses on the object detection task of the COCO dataset consisting in localizing the objects in an image with bounding boxes and categorizing each one of them between 80 categories. The dataset changes each year but it is usually composed of more than 120,000 images for training and validation, and more than 40,000 images for testing.

Evaluation metrics. The object detection challenge contains a regression and a classification task. First of all, the bounding boxes with low confidence (the model usually outputs many more boxes than actual objects) are removed to assess the spatial precision. Then, the Intersection over Union (IoU) is defined as the percentage $\frac{|A \cap B|}{|A \cup B|}$, where A is the area of the predicted bounding box and B is the ground-truth box. The higher the IoU, the better the predicted location of the box for a given object. A threshold is usually applied on the IoU values to select the bounding box candidates.

The mean Average Precision (mAP) metric consists in computing the Average Precision (AP) over all the categories of the dataset. Let $\text{AP} = \frac{\text{TP}}{\text{TP+FP}}$ be the Average Precision (AP) with TP the true positive and FP the false positive predictions, we note $\text{mAP} = \frac{1}{K} \sum_{k=0}^{K-1} \text{AP}_k$ where K is the number of classes in the dataset and AP_k is the AP for the k-th class. The mAP metric avoids extreme specialization in a few classes and thus weak performances in the others. This metric considers only the predicted bounding boxes with a sufficient overlap with the ground truth, *i.e.* with a threshold on the IoU. The IoU threshold is usually fixed but a high number of bounding boxes increases the number of candidate boxes. The COCO challenge has developed an official metric avoiding an over-generation of boxes. It computes a mean of the mAP scores for a set of IoU threshold values in order to penalize a high number of bounding boxes with wrong classifications. This set of thresholds is defined between 0.5 and 1 with a 0.05 step.

Region-based Convolutional Network (R-CNN). This method starts with a selective search [Uijlings *et al.* 2013] initializing small regions in an image and merging them with a hierarchical grouping. The detected regions are merged according to a variety of color spaces and similarity metrics. It outputs a small number of region proposals which could contain an object. The R-CNN model [Girshick *et al.* 2016] combines the selective search method detecting region proposals and deep learning to classify the objects. Each region proposal is resized to match the input of a CNN outputting a 4096-dimension vector of features. This vector is then used as input of binary SVM [Hearst *et al.* 1998] classifiers, one for each class. It is also used as input of a linear regressor adapting the shapes of the corresponding bounding box to reduce the location error. The CNN is trained on the 2012 ImageNet dataset for classification. It is then fine-tuned using the region proposals corresponding to an IoU greater than 0.5 with the ground-truth boxes. Two versions are produced, one version is using the 2012 PASCAL VOC dataset and the other the 2013 ImageNet dataset with bounding boxes. The SVM classifiers are also trained for each class of each dataset. The best R-CNN models have achieved a 62.4% mAP score on the 2012 PASCAL VOC challenge (22.0 points increase w.r.t. the second best result on the leader

board) and a 31.4% mAP score over the 2013 ImageNet dataset (7.1 points increase w.r.t. the second best result on the leader board).

Fast Region-based Convolutional Network (Fast R-CNN). The objective of Fast R-CNN [Girshick 2015] is to reduce the computational cost due to the diversity of models required to analyse all region proposals. A CNN takes the entire image as input instead of a specialised one for each region proposal (R-CNN). Region of Interests (RoIs) are detected with the selective search method applied on the produced feature maps. Formally, the feature maps size is reduced using a RoI pooling layer to get valid Region of Interests with fixed height and width as hyperparameters. Each RoI layer feeds fully connected layers[12] creating a vector of features. The vector is used to predict the observed object with a softmax classifier and to adapt the bounding box location with a linear regressor. The best Fast R-CNN models have reached mAP scores of 70.0% for the 2007 PASCAL VOC challenge, 68.8% for the 2010 PASCAL VOC challenge and 68.4% for the 2012 PASCAL VOC challenge.

Faster Region-based Convolutional Network (Faster R-CNN). The aim of the Faster R-CNN [Ren *et al.* 2015] is to replace the selective search method with a Region Proposal Network (RPN) to generate region proposals, predict bounding boxes and detect objects only with CNNs. The Faster R-CNN combines an RPN and a Fast R-CNN model.

The RPN takes as input the entire image and produces feature maps. A window of size 3×3 slides over all the feature maps and outputs a 256-dimension vector of features linked to two fully connected layers, one for box regression and one for box classification. Multiple region proposals are predicted by the fully connected layers. A maximum of k regions is fixed, thus the output of the box regression layer has a size of $4k$ (for each box, coordinates of a corner of the boxes and its height and width) and the output of the box classification layer a size of $2k$ ("objectness" scores to detect an object or not in the box). The k region proposals detected by the sliding window are called anchors. When the anchor boxes are detected, they are selected with a threshold applied on the "objectness" score keeping only the relevant boxes. These anchor boxes and the feature maps computed by the initial CNN model are used as input of a Fast R-CNN model.

Faster R-CNN uses RPN to avoid the selective search method, it accelerates the training and testing processes while improving the performances. The RPN is a pre-trained model using the ImageNet dataset for classification, it is then fine-tuned on the PASCAL VOC dataset. The best Faster R-CNN models have obtained mAP scores of 78.8% on the 2007 PASCAL VOC challenge and 75.9% on the 2012 PASCAL VOC challenge. The models have been trained with PASCAL VOC and COCO datasets. One of these models[13] is 34 times faster than the Fast R-CNN using the selective search method.

[12]The entire architecture is inspired from the VGG16 model, thus it has 13 convolutional layers and 3 fully connected layers.

[13]The fastest Faster R-CNN has an architecture inspired by the ZFNet model introduced by [Zeiler & Fergus 2014]. The commonly used Faster R-CNN has an architecture similar to the VGG16 model and it is only 10 times faster than the Fast R-CNN.

You Only Look Once (YOLO). The YOLO model [Redmon *et al.* 2016] is a single stage approach predicting bounding box coordinates and class probabilities with a single network. Its simplicity permits real time predictions during inference. The model takes an image as input which is divided in an $S \times S$ grid. In each cell of this grid is predicted B bounding boxes with a confidence score. This confidence is defined as the probability to detect the object multiply by the IoU between the predicted and the ground-truth boxes. The architecture is inspired by the GoogLeNet model [Szegedy *et al.* 2015] using inception modules. It has 24 convolutional layers followed by 2 fully-connected layers. The inception modules are replaced by 3×3 followed by 1×1 convolutions. The final layer outputs a tensor of dimensions $S \times S \times (K + B \times 5)$ corresponding to the predictions in each cell of the grid, where K is the number of classes, B the fixed number of anchor boxes per cell, each anchor being characterized with 4 coordinates (coordinates of the center of the box, width and height) and a confidence value. The YOLO model predicts a high number of bounding boxes since it does not localize Region of Interest (RoI). Thus, there are a lot of bounding boxes without any object. Non-Maximum Suppression (NMS) it is applied at the end of the network. It consists in merging highly-overlapping bounding boxes of a same object into a single one. The YOLO model reached a 63.7% mAP score on the 2007 PASCAL VOC challenge and a 57.9% mAP score over the 2012 PASCAL VOC challenge. A few years later, the YOLO architecture has been extended in YOLOv2 and the authors proposed a new model, YOLO9000, capable of detecting more than 9000 categories while running in almost real time (around 10 Frames-per-Second (FPS)). Details are provided in Appendix A.1.1.

Single-Shot Detector (SSD). The SSD [Liu *et al.* 2016] is a single stage model, similarly to YOLO, predicting the bounding box coordinates and the class probabilities simultaneously. This end-to-end architecture is fully convolutional using different kernel sizes (10×10, 5×5 or 3×3). Feature maps from convolutional layers at different levels of the network are used to predict the bounding boxes. The feature maps are processed by a specific convolutional layers with a 3×3 kernel called extra feature layers producing a set of bounding boxes similar to the anchor boxes of the Fast R-CNN. Each box has 4 parameters: the coordinates of its center, its width and its height. At the same time, a branch produces a vector of probabilities corresponding to the softmax considering all the classes of object. The NMS method selecting the relevant bounding boxes is also used at the end of the SSD model. Hard Negative Mining (HNM) selects relevant boxes among the negative samples during training: the boxes with the highest confidence are selected depending on the ratio between the negative and the positive samples (evaluated to $\frac{1}{3}$ in this work). In their work, [Liu *et al.* 2016] distinguished the SSD300 and the SSD512, the latter corresponds to a SSD300 with an extra convolution on the prediction heads. The best SSD models are trained with the 2007, 2012 PASCAL VOC datasets and the 2015 COCO dataset with data augmentation. They obtained mAP scores of 83.2% on the 2007 PASCAL VOC challenge and 82.2% on the 2012 PASCAL VOC challenge. On the 2015 COCO challenge, they reached a score of 48.5% for an IoU threshold of 0.5, 30.3% for an IoU threshold of 0.75 and 31.5% for the official mAP metric.

Mask Region-based Convolutional Network (Mask R-CNN). An extension of the Faster R-CNN has been proposed by [He *et al.* 2017] performing simultaneously object detection and semantic segmentation as a multi-task problem. It uses the Faster R-CNN pipeline with three output branches for each candidate object: a class label, a bounding box offset and the object mask. It uses RPN to generate bounding box proposals and produces the three outputs at the same time for each RoI. The initial RoIPool layer, used in the Faster R-CNN to select object proposals in the feature maps, is replaced by a RoIAlign layer. It removes the quantization of the coordinates of the original RoI and computes the exact coordinates of the locations. The RoIAlign layer provides scale-equivariance and translation-equivariance with the region proposals. The backbone extracting the features is a ResNeXt architecture [Xie *et al.* 2017] with 101 layers. Each residual block is slightly modified from the work of [He *et al.* 2016] by considering multiple parallel convolutions producing feature maps which are stacked and followed by a residual connection. The model detects RoIs which are processed with a RoIAlign layer. One branch of the network is linked to a fully connected layer adjusting the coordinates of the bounding boxes and predicting the class probabilities. The other branch is linked to two convolutional layers, the last one computes the mask of the detected object. Additional details about semantic segmentation are provided in the next section. The multi-task training is performed by summing the loss function corresponding to each task into a global one. The gradients are propagated in the entire network. The Mask R-CNN outperformed the state of the art in the four COCO challenges: the instance segmentation, the bounding box detection, the object detection and the key point detection. It reached mAP scores of 62.3% with an IoU threshold of 0.5, 43.4% for an IoU threshold of 0.75 and 39.8% for the official metric over the 2016 COCO challenge.

Conclusion. Through the years, object detection models tend to infer localisation and classification all at once to have an entirely differentiable network. They can be trained from head to tail with back-propagation. However, a trade-off between high performance and real time prediction capability is made in the last presented models. This review of object detection methods using deep learning, supplemented by Appendix A.1.1, is not exhaustive and ranges from 2015 to 2017. Recent advances have largely extended these approaches but the previously presented ones are still commonly used.

2.6.3 Semantic segmentation

The semantic segmentation task applied to natural images consists in classifying each pixel in a category. An extension of this task, called instance segmentation, consisting in classify each pixel in a category with an identification number to distinguish different instances of the same category.

Most of the object detection pipelines presented in Section 2.6.2 require anchor boxes or proposals to localize an object in a scene. Unfortunately, just a few models take into account the entire context of an image. They are still limited on localizing objects with small part of the information. They cannot provide a full comprehension of a scene.

Scene understanding requires high visual perception of each entity while considering

the spatial information. In the past few years, other challenges have emerged to better understand the actions in a image or a video: key point detection, action recognition, video captioning, visual question answering and so on. This section will focus on the semantic segmentation task. It will introduce datasets and evaluation metrics; and detail several well-known architectures and methods. This list is not exhaustive and additional methods are detailed in Appendix A.1.1.

Datasets. The PASCAL VOC dataset[14] (2012) [Everingham *et al.* 2015], mentioned in the previous section, is well-known and commonly used for object detection and segmentation covering 20 categories. More than 11,000 images compose the train and validation datasets while 10,000 images are dedicated to the test dataset.

The PASCAL-Context dataset[15] (2014) [Mottaghi *et al.* 2014] is an extension of the 2010 PASCAL VOC dataset. It contains around 10,000 images for training, 10,000 for validation and 10,000 for testing. The specificity of this release is that the entire scenes are segmented between more than 400 categories. Note that the images have been annotated during three months by six in-house annotators.

There are two COCO challenges[16] (in 2017 and 2018) for image semantic segmentation: "object detection" and "stuff segmentation". The object detection task consists in segmenting and categorizing objects into 80 categories. The stuff segmentation task consists in segmenting almost the entire visual information of in an image, *e.g.* including sky, wall, grass. This section will denote the COCO semantic segmentation challenge as the corresponding object detection task since it is the widely explored of the two. The COCO dataset [Lin *et al.* 2014b] for semantic segmentation is composed of more than 200,000 images with over 500,000 object instances segmented which are splitted in train, validation and test[17]. These datasets contain 80 categories and only the corresponding objects are segmented.

The Cityscapes dataset[18] [Cordts *et al.* 2016] is composed of complex segmented urban scenes from 50 cities. There are around 23,500 images for training and validation (fine and coarse annotations) and 1,500 images for testing (only fine annotation). The images are fully segmented as the PASCAL-Context dataset with 29 classes, within 8 super categories: flat, human, vehicle, construction, object, nature, sky, void. This dataset is well-known for its semantic segmentation task because of its complexity and its similarity with real urban scenes for autonomous driving applications.

Evaluation metrics. The semantic segmentation task is commonly evaluated using the mean Intersection over Union (mIoU) metric. As presented in Section 2.6.2, the IoU quantifies the ratio between the overlapping area and the union area between a predicted shape

[14] http://host.robots.ox.ac.uk/pascal/VOC/

[15] https://cs.stanford.edu/~roozbeh/pascal-context/

[16] http://cocodataset.org

[17] As in many challenges, the test dataset is divided in test-dev (for research) and test-challenge (for the challenge). The annotations for both datasets are not available

[18] https://www.cityscapes-dataset.com/

and its ground truth. The mIoU is the average of IoU of all the predicted shapes considering their ground truth over all the classes.

The official evaluation metric of the Pascal-VOC, PASCAL-Context and Cityscapes challenges is the mIoU. The COCO challenge also considers the mAP and mean Average Recall (mAR) metrics for evaluation. Differently from the object detection task, these two metrics are computed pixel-wise without filtering the shapes with an IoU threshold. The mAR metric is computed similarly than the mAP detailed in Section 2.6.2. Let $\text{AR} = \frac{\text{TP}}{\text{TP+FN}}$ be the Average Recall (AR) with TP the true positive and FN the false negative predictions, we note $\text{mAR} = \frac{1}{K}\sum_{k=0}^{K-1} \text{AR}_k$ where K is the number of classes in the dataset and AR_k is the AR for the k-th class. The works detailed in this section refer to their performances in term of AP and AR but these metrics are equivalent to the presented mAP and mAR respectively. We chose to keep the same notations than in Section 2.6.2 for easier understanding.

Fully Convolutional Network (FCN). The Fully Convolutional Network (FCN) proposed by [Long *et al.* 2015] is the first model composed only of convolutional layers trained end-to-end with back-propagation for image segmentation. FCNs are used to learn features at different scales. It processes an input image with convolutions and down-samplings. Fully connected layers of well-known architectures (AlexNet, VGG16, GoogLeNet) are replaced by convolutions to allow non-fixed size inputs. The convolutions output feature maps with lower and lower dimensions. Thus, they are up-sampled with up-convolutions (the stride factor is inferior to 1) to recover the original input size. The FCN uses skip connections from features learnt at different levels of the network to generate the final output. We denote with FCN-32s a network where the output mask is generated only by up-sampling and processing feature maps with $1/32$ resolution of the input. Similarly, FCN-16s is a network where $1/32$ and $1/16$ resolution features maps are used to generate the output mask. In the same manner, FCN-8s fuses $1/32$, $1/16$ and $1/8$ resolution feature maps for output prediction. This way, the model classifies each pixel of the input into a category and it is trained using a pixel-wise loss. The FCN-8s reached a 62.2% mIoU score on the 2012 PASCAL VOC segmentation challenge using pre-trained models on the 2012 ImageNet dataset.

U-Net. [Ronneberger *et al.* 2015] proposed an extension of the FCN [Long *et al.* 2015] for biological microscopy images. The U-net architecture is composed of two pathways: a contracting pathway to compute features and an expanding pathway to spatially localise patterns in the image. The down-sampling or contracting pathway has a FCN architecture extracting features with 3×3 convolutions. The up-sampling or expanding pathway uses up-convolution (or deconvolution as detailed in Appendix A.1.2) reducing the number of feature maps while increasing their dimensions (height and width). Cropped feature maps from the down-sampling pathway of the network are copied and stacked within the up-sampling pathway to avoid loosing information. Finally, a 1×1 convolution processes the feature maps to generate a soft mask of probability to categorise each pixel of the input image. Since then, the U-net architecture has been widely extended in recent works, a

few of them will be presented in the next paragraphs (FPN, PSPNet, DeepLabv3). This architecture does not use any fully connected layer. As a consequence, the number of parameters of the model is reduced and it can be trained with a small labelled dataset (using appropriate data augmentation). *E.g.* the authors have used a public dataset with 30 images in their experiments.

Feature pyramid network (FPN). The Feature Pyramid Network (FPN) developped by [Lin *et al.* 2017a], inspired from U-Net [Ronneberger *et al.* 2015], is used in object detection or image segmentation frameworks. The architecture is composed of a bottom-up pathway, a top-down pathway and lateral connections in order to join low-resolution and high-resolution features. The bottom-up pathway takes an image with an arbitrary size as input. It is processed with convolutional layers and down-sampled by max-pooling layers. Note that each group of feature maps with the same size is called a stage. The output feature maps of the last layer at each stage are used for the lateral connections of the feature pyramid. The top-down pathway consists in up-sampling the last feature maps with un-pooling while enriching them with feature maps from the same stage of the bottom-up pathway using lateral connections. Each lateral connection processes the feature maps of the bottom-up pathway with a 1×1 convolution (reducing the number of feature maps) and stacks its output with the un-pooled feature maps of the top-down pathway. The concatenated feature maps are then processed by a 3×3 convolution producing the output of the stage. Finally, each stage of the top-down pathway generates a prediction to detect an object. For semantic segmentation, the authors use two fully connected layers to generate two masks with different sizes over the objects. It works similarly to Region Proposal Network with anchor boxes (see R-CNN [Girshick *et al.* 2016], Fast R-CNN [Girshick 2015], Faster R-CNN [Ren *et al.* 2015]). According to the authors, the efficiency of this architecture is due to an improvement in the propagation of the information from the last layers of in the network. The FPN based on DeepMask ([Pinheiro *et al.* 2015]) and SharpMask ([Pinheiro *et al.* 2016]) frameworks achieved a 48.1% mAR score on the 2016 COCO segmentation challenge.

Mask R-CNN. As presented in 2.6.2, the Mask R-CNN [He *et al.* 2017] consists in a Faster R-CNN with three output branches. First, it uses an RPN extracting RoI and features are learnt with the RoIPool layer. Then the output branches are specialized in computing the bounding box coordinates, predicting the associated class and the binary mask[19] to segment the object. The binary mask has a fixed size and it is generated by a FCN for each RoI. It also uses a RoIAlign layer instead of a RoIPool to avoid misalignments due to the quantization of the RoI coordinates. The particularity of the Mask R-CNN model is its multi-task loss combining the losses of the bounding box coordinates, the predicted class and the segmentation mask. The model tries to solve complementary tasks leading to better performances on each individual task. The best Mask R-CNN uses a ResNeXt [Xie *et al.* 2017] to extract features and an FPN architecture. It has obtained a 37.1% mAP

[19]The Mask R-CNN model computes a binary mask for each object for a predicted class (instance-first strategy) instead of classifying each pixel into a category (segmentation-first strategy).

score on the 2016 COCO segmentation challenge and a 41.8% mAP score on the 2017 COCO semantic segmentation challenge.

Atrous convolutions (DeepLab and extensions). Inspired by the FPN model of [Lin *et al.* 2017a], [Chen *et al.* 2018a] proposed the DeepLab architecture combining atrous convolution, spatial pyramid pooling and fully connected CRFs. This model is also called the DeepLabv2, it is an adjustment of the original DeepLab model which will not be detailed here to avoid redundancy. The authors have introduced the atrous convolution which is equivalent to the dilated convolution of [Zhao *et al.* 2017]. It consists of convolutions with sparse kernels targeting spread pixels. The number of neurons in the kernel does not change but they are separated by a fixed number of pixels, named dilation rate. The atrous convolution helps to capture multiple scales of objects. When it is used without max-pooling, it increases the resolution of the final output without increasing the number of parameters. The Atrous Spatial Pyramidal Pooling (ASPP), inspired by PSPNet [Zhao *et al.* 2017] (see Appendix A.1.2), consists in applying parallel atrous convolutions using the same input with different dilation rates. The features maps are processed in separate branches and concatenated using bilinear interpolation to recover the original size of the input. The ASPP module helps to learn patterns at different scales with different receptive fields due to the various dilation rates used. The feature maps are then processed by a fully connected Conditional Random Field (CRF) [Krähenbühl & Koltun 2011] computing edges between the features and long term dependencies to produce the semantic segmentation. The best DeepLab using a ResNet-101 [He *et al.* 2016] as backbone has reached a 79.7% mIoU score on the 2012 PASCAL VOC challenge, a 45.7% mIoU score on the PASCAL-Context challenge and a 70.4% mIoU score on the Cityscapes challenge.

In their work, [Chen *et al.* 2017] have revisited the DeepLab framework to create DeepLabv3 combining cascaded and parallel modules of atrous convolutions. The authors have modified the ResNet architecture to keep high resolution feature maps in deep blocks using atrous convolutions. An ASPP module is used with an additional 1×1 layer rand batch normalization. The contenated outputs are processed by a final 1×1 convolution to predict the segmentation masks. The best DeepLabv3 model with a ResNet-101 pretrained on ImageNet and JFT-300M [Sun *et al.* 2017] datasets has reached 86.9% mIoU score in the 2012 PASCAL VOC challenge. It also achieved a 81.3% mIoU score on the Cityscapes challenge with a model only trained with the associated training dataset.

The final version called DeepLabv3+ [Chen *et al.* 2018b] uses an encoder-decoder structure. The author first introduce atrous separable convolution composed of a depth-wise convolution (spatial convolution for each channel of the input using a dilation rate > 1) and point-wise convolution (1×1 convolution with the depth-wise convolution as input). The DeepLabv3 architecture has been used as encoder. The authors extracted features with a Xception [Chollet 2017] architecture with modifications: they added convolutional layers, they replaced max-pooling layers by atrous depth-wise separable convolutions, and they added batch normalization and ReLU activation following each 3×3 convolutions. The feature maps of the backbone are processed by an ASPP module with a final 1×1 convolution reducing the number of maps while up-sampling them with a factor of four.

The decoder processes the feature maps from the backbone and from the ASPP module with convolutions. The final maps are up-sampled by a factor of four to recover the input dimension and produce the segmentation masks. The best DeepLabv3+ pre-trained on the COCO and the JFT-300M [Sun *et al.* 2017] datasets obtained a 89.0% mIoU score on the 2012 PASCAL VOC challenge. The model trained on the Cityscapes dataset reached a 82.1% mIoU score for the corresponding challenge.

Conclusion. The section, supplemented by Appendix A.1.2, described a non-exhaustive list of neural network architectures for semantic segmentation published between 2015 and 2018. A recurrent problem is these approaches is the lack of global visual context in the features learnt by the network, partially explored in the EncNet architecture (see Appendix A.1.2). The state of the art methods used multiple pathways in the network to better propagate the information and to learn relations between the objects.

Considering an entire image, pixel-wise predictions allow a more accurate understanding of a scene and its environment. It is especially true in the context of autonomous driving in complex urban scenes. Multi-task learning has also shown that neural networks are able to learn more relevant features by solving complementary tasks at the same time. As an improvement, a multi-head architecture could be considered to solve the semantic segmentation task with other tasks proposed in the COCO challenge (*e.g.* key point detection, action recognition, video captioning or visual question answering). The following section will briefly introduce deep learning methods and architectures for 3D point cloud scene understanding.

2.6.4 Methods for 3D point clouds

Scenes can also be represented in three dimensions, including 3D point clouds, in order to obtain accurate geometric information. These are points located in 3D Cartesian coordinates. The point cloud representation is sparse, which means that it does not completely fill the Cartesian space, unlike an image composed of dense adjacent pixels. Methods presented in Sections 2.6.1, 2.6.2 and 2.6.3 cannot be directly applied to a point cloud representation as convolutional layers process dense inputs.

The tasks presented in the previous sections for scene understanding have their own equivalence when applied to point clouds: classification of an entire point cloud, prediction of bounding boxes and point-wise semantic segmentation. In the automotive industry, the LiDAR sensor is commonly used for theses tasks since it provides 3D point clouds recordings. Processed RADAR data can be represented as a point cloud in a Cartesian space (see Section 2.2). The low resolution of the elevation angle of a LD RADAR leads to process the RADAR point cloud as a Bird's Eye View (BEV) representation, but both representations are generally processed with similar methods.

This section introduces well-known deep neural network architectures specialised in point cloud processing. It aims to provide insights for a better comprehension of the related work on RADAR point clouds that will be presented in Chapter 3.

PointNet. The PointNet model proposed by [Qi *et al.* 2017a] is the first end-to-end neural network architecture that processes unordered sets of 3D points while being adapted for point cloud classification, point-wise semantic segmentation and scene semantic parsing. Its first module processes the point cloud with a mini-network called T-Net, based on successive fully connected layers extracting point-wise features. The final layer of the T-Net is a MLP, shared between all the points, predicting an affine transformation matrix. This matrix is directly applied to the coordinates of the input point cloud to adjust its geometric representation. Then, a shared MLP learns point-wise features from the transformed point cloud and forwards them to the next stage. Successive T-Net transformation and shared MLP process the representations before applying a max-pooling on the point dimension producing a global feature vector. This vector is then used either as input of an MLP of point cloud classification, or in a segmentation network for semantic segmentation.

The segmentation network combines local features from the middle stage MLP to the global feature vector. The concatenated representations are then processed by two successive MLPs shared between all the points: the first combines the local and global concatenated features, the second predicts a class for each point to perform semantic segmentation.

The PointNet architecture uses only shared MLPs and fully connected layers to learn local and global features from an unordered point cloud. Moreover, it can easily be adapted to the application. This method reached the best performances for point cloud classification, segmentation and scene parsing with less computational complexity than competing methods. To this day, PointNet is still used as a backbone for point cloud feature extraction for both LiDAR and RADAR point clouds.

PointNet++. An extension named PointNet++ [Qi *et al.* 2017b] has been proposed to capture local patterns with increasing contextual scales in order to improve performances in complex scenes. The authors proposed a hierarchical neural network applying PointNet recursively to increase the receptive field of each layer similarly to stacked convolutions for image processing. This network learns fine-grained structures and either uses the last down-scaled point cloud as input of a MLP for classification, or merges feature to obtain the global context of the scene for segmentation.

Additionally to the recursive PointNet, the authors proposed the Multi Scale Grouping (MSG) and the Multi Resolution Grouping (MRG) approaches. The MSG operates before the PointNet to concatenate features from different scale levels and obtain a multi-scale feature. The MRG initially aims to apply a local PointNet on each point provided by the previous layer. This method is computationally expensive, so the authors approximated it by using a single PointNet applied to each local group of features which have been aggregated at a higher level. For the segmentation task, they proposed a Point Feature Propagation to segment the original point cloud from a down-sampled point cloud. Each classified point is interpolated at the upper level using an inverse weighting average of its local neighborhood while the skip connections of the network transmit the information to each level.

The PointNet++ succeeds to learn hierarchical features with a recursive PointNet while fusing multi-scale features with grouping approaches. The proposed architecture is more

computationally expensive than PointNet, but it provides better performance and can also be adapted to various applications. It is also commonly used for 3D scene understanding depending on the trade-off to reach between the performances and the inference time.

Pixor. The PIXOR architecture [Yang *et al.* 2018] has been introduced as a single stage method for 3D object detection from LiDAR point cloud claiming real time inference useful for autonomous driving. The authors transformed a 3D point cloud in dense 2D BEV representations[20], each one corresponding to an elevation angle discretization. The 2D BEVs are transformed in occupancy grids with fixed resolutions and stacked in the channel dimension.

The PIXOR architecture is composed of an FPN backbone with successive 2D convolutions and down-sampling layers while respecting that the last feature map has sufficient resolution per pixel to contain an object. The head part of the network processes the backbone feature maps and performs object recognition and location simultaneously. The location task consists of classifying each pixel of the feature map as being an object or not. The recognition task regresses 6 parameters per pixel corresponding to the offset of the object's position from the pixel location, its size and orientation.

The proposed architecture uses 2D convolutions on 3D points clouds transformed in dense representations while being a single stage approach. It reached state of the art performances in 3D object detection while operating in real time and is therefore a good trade-off for autonomous driving applications.

Conclusion. Point clouds are sparse representations that need to be either processed point-wise, or transformed in dense representations to use conventional methods. This section provided details on the most mainstream deep learning architectures for point cloud processing, namely PointNet, PointNet++ and PIXOR. Many recent methods and specific sparse convolution operations have been designed for 3D point cloud processing but they are out of the scope of this thesis. We suggest that the reader refers to the work of [Guo *et al.* 2020] for in-depth information.

The following chapter will review the related work on RADAR datasets and deep learning methods applied to RADAR data for scene understanding, in particular for classification, object detection and semantic segmentation.

[20] One of the advantages of the BEV representation is that objects do not overlap w.r.t. the front-views.

CHAPTER 3

Related work

Contents

3.1 Diverse applications . . . 41
3.2 Automotive RADAR datasets . . . 43
3.2.1 Traditional RADAR . . . 44
3.2.2 Scanning RADAR . . . 45
3.2.3 High-definition RADAR . . . 45
3.2.4 Our proposals . . . 46
3.3 RADAR object detection . . . 46
3.3.1 Range-Angle-Doppler tensor . . . 47
3.3.2 Range-Angle or Range-Doppler view . . . 47
3.3.3 RADAR point cloud . . . 48
3.4 RADAR semantic segmentation . . . 49
3.4.1 Range-Angle view . . . 50
3.4.2 RADAR point cloud . . . 50
3.5 Sensor fusion . . . 51
3.5.1 RADAR and camera fusion . . . 51
3.5.2 RADAR and LiDAR fusion . . . 53
3.5.3 RADAR, camera and LiDAR fusion . . . 55
3.6 Conclusions . . . 55

Over the years, deep learning methods have evolved to suit many research areas. This chapter reviews the related work of deep learning algorithms applied to RADAR data. Section 3.1 briefly introduces diverse applications using RADAR data. In Section 3.2, we focuses on open source RADAR datasets, which are a prerequisite for improvements in deep learning algorithms. Sections 3.3 and 3.4 respectively review the related works on object detection and semantic segmentation applied to automotive RADAR. Finally, Section 3.5 reports on the applications of sensor fusion including a RADAR sensor.

3.1 Diverse applications

As detailed in Section 2.1, a RADAR sensor provides the position and Doppler of the surrounding reflectors, which is interesting to explore in many research areas. Unlike with a camera, it is difficult to identify and distinguish people in RADAR data. Therefore this

sensor helps to understand scenes while respecting the privacy of the users. In particular, hand gesture recognition using RADAR data has been widely explored because of its application to human control on a mobile device, an electronic watch or inside a car.

Classification of hand gestures has been explored with end-to-end CNN applied to Doppler spectrograms[1] [Kim & Toomajian 2016, Dekker *et al.* 2017]. Simulated Doppler spectrograms have also been used to train an MLP for gesture classification [Ishak *et al.* 2018]. In their work, [Wang *et al.* 2016] used CNNs to generate embeddings of RD sequences used as input of an RNN with an LSTM cell. The presented results have highlighted that the temporal information is relevant for a classification task using RADAR data. In the same way, [Zhang *et al.* 2018b] classified sequences of range spectrograms stacked in the temporal dimension with 3D convolutions and an LSTM cell trained with a CTC loss[2] function [Graves *et al.* 2006]. Unsupervised learning has also been explored to learn representations of range spectrograms with VGG AutoEncoders [Zhang *et al.* 2019], which are then fine-tuned for hand gesture classification. [Lei *et al.* 2020] suggested an architecture with different branches to process Range-Doppler and Range-Angle views with an RNN exploiting the temporal axis for hand gesture classification. An additional spatio-Doppler attention has also been proposed by [Hazra & Santra 2019]. Since objects' signatures evolve through time, [Scherer *et al.* 2021] proposed an architecture for hand gesture classification with a CNN backbone and a Temporal Convolutional Network (TCN) using dilated 1D convolutions extracting temporal features from RD and achieving better results than RNN with LSTM. The RAD tensor has also been considered for hand gesture classification by being aggregated in views which are processed by separate branches and finally merged [Wang *et al.* 2021c]. [Sun *et al.* 2019] detected multiple gestures with a Faster R-CNN including 3D convolutions on Doppler spectrograms recorded inside a car. Segmentation and detection of several gestures have also been explored by improving Mask R-CNN with additional modules of max and average pooling processed by MLPs [Wang *et al.* 2019]. Indoor car driver's hand gesture recognition has been tackled with multi-class classification using sensor fusion with camera, depth estimation and Doppler spectrograms processed with 3D convolutions [Molchanov *et al.* 2015].

Deep learning models have also shown interesting results in differentiating human activities. Doppler spectrograms have been used for human detection and indoor activity classification [Kim & Moon 2016]. A more advanced architecture has been proposed by [Zhu *et al.* 2020] using fully 1D convolutions and RNN with LSTM cells to solve the same task. Doppler spectrograms have been explored for human gait classification including the temporal information with RNN and LSTM cells [Klarenbeek *et al.* 2017] or more advanced architectures with dual stream of the Doppler representation [Chen *et al.* 2021b] based on Vision Transformers (ViT) [Dosovitskiy *et al.* 2021]. Deformable convolutions [Dai *et al.* 2017] have been used to adapt convolutional kernels to objects' signature on Doppler spectrograms for fall motion classification.

RADAR sensors have also been exploited for outdoor application, *e.g.* to classify

[1] A Doppler spectrogram representation is a time-Doppler representation corresponding to a single FFT applied on the recorded RADAR data considering the chirp sampling axis (see Section 2.2)

[2] The Connectionist Temporal Classification (CTC) function [Graves *et al.* 2006] is specialized in calculating a loss between a continuous time series and a target sequence.

Unmanned Aircraft Vehicles (UAV) using a temporal FCN on Doppler spectrograms [Brooks *et al.* 2018]. [Wang *et al.* 2020b] explored ice layer segmentation with a ground-penetrating RADAR. Counting people with RADAR data has been addressed by adapting a network trained on natural images to RA RADAR views and Doppler spectrograms using knowledge distillation [Aydogdu *et al.* 2020]. Since RADAR representations contain speckle noise, denoising AutoEncoders have been explored on RD views [de Oliveira & Bekooij 2020] using filtered representations as ground truth. Micro-motions of raw RADAR data in the temporal domain have been used as input of an AutoEncoder to estimate contactless electrocardiograms while considering domain transformation [Chen *et al.* 2021a]. Unsupervised learning has also been explored from RA map of a scanning RADAR to learn feature embeddings using a deep learning architecture for place recognition [Gadd *et al.* 2021].

The Synthetic Aperture RADAR (SAR) is used as an embedded sensor on orbital satellites, planes and aerial platforms. It exploits the motion of the RADAR's antennas to create an image and uses the combination of multiple pulses to create a synthetic aperture and improve its spatial resolution. SAR data are similar to 2D BEV representations, they have been used in many applications, *e.g.* for pixel-wise land segmentation [Zhang *et al.* 2017b, Zhang *et al.* 2020, Pham & Lefevre 2021] or sensor fusion for panoptic semantic segmentation [Garnot *et al.* 2021]. Deep learning applications have been developed for SAR data, in particular for earth observation, but are beyond the scope of this thesis.

The lack of open source RADAR data (with or without annotation) has impeded deep learning research in many fields. Generating data as realistic Doppler spectrograms of simple objects using Generative Adversarial Networks (GANs) [Goodfellow *et al.* 2014] has been explored [Truong & Yanushkevich 2019]. Synthetic RADAR data have been simulated using CycleGAN [Zhu *et al.* 2017] for ice layer segmentation [Rahnemoonfar *et al.* 2020]. Doppler spectrograms have also been generated using GANs for human gait classification. The automotive industry also suffers from the lack of open source RADAR dataset. Generative approaches could address this issue but they have shown limitations to reproduce the properties of RADAR data (see Section 4.2.1). The following section will detail existing automotive RADAR datasets specialized in scene understanding.

3.2 Automotive RADAR datasets

Deep learning algorithms, trained with large amounts of annotated data, reached impressive performances for scene understanding tasks. Annotated datasets are required to drive research in scene understanding using RADAR representations. This section will review the existing open source datasets for RADAR scene understanding which have been released over the past three years.

Dataset	Year	Scale	RADAR data					RADAR type	Modalities	Sequence	Annotation type
			ADC	RAD	RA or RD	PC	Doppler				
nuScenes	2019	Large	✗	✗	✗	✓	✓	LD	CLO	✓	3D Boxes
Astyx	2019	Small	✗	✗	✗	✓	✓	HD	CL	✗	3D Boxes
RadarRobotCar	2020	Large	✗	✗	✓	✗	✗	S	CLO	✓	✗
RADIATE	2020	Medium	✗	✗	✓	✗	✗	S	CLO	✓	2D Boxes
MulRan	2020	Medium	✗	✗	✓	✓	✗	S	CLO	✓	✗
Zendar	2020	Small	✗	✗	✓	✓	✓	HD	CL	✓	2D Boxes
CARRADA (Sec. 4.3)	2020	Small	✗	✓	✓	✓	✓	LD	C	✓	2D Boxes, Seg.
CRUW	2021	Medium	✗	✗	✓	✗	✗	LD	C	✓	Point Location
RadarScenes	2021	Large	✗	✗	✗	✓	✓	HD	CO	✓	Point-wise
RADDet	2021	Small	✗	✓	✓	✗	✓	LD	C	✓	2D Boxes
RADIal (Chap. 6)	2021	Medium	✓	✓	✓	✓	✓	HD	CLO	✓	2D Boxes, Seg.

Table 3.1: **Publicly-available driving RADAR datasets**. The dataset scale is Small ($<$ 15k frames), Large ($>$ 130k frames) or Medium (in between). The used RADAR is LD, HD or Scanning (S). RADAR data are released in different representations, amounting to different signal processing pipelines: ADC signal, RAD tensor, RA view, RD view, Point Cloud (PC). Presence of Doppler information depends on the RADAR sensor. Other sensor modalities are Camera (C), LiDAR (L) and Odometry (O). CARRADA (see Section 4.3) is the only dataset providing both RAD and PC of a LD RADAR with bounding box and segmentation annotations. RADIal (see Chapter 6) is the only dataset providing each representation of a HD RADAR, combined with camera, LiDAR and odometry, while proposing detection and free space segmentation tasks.

3.2.1 Traditional RADAR

Traditional RADARs, or LD RADARs, offer a good trade-off between cost and performance. Most of them are composed of two transmitter and four receiver antennas, leading to eight virtual antennas. They provide accurate range and velocity while being robust to adverse weather conditions. The **nuScenes** dataset [Caesar *et al.* 2020] is composed of 5.5 hours of recorded sequences in two countries including night and rain weather conditions. This dataset is composed of simultaneous recordings from five RADARs. It has an additional 32-beam LiDAR and 6 cameras. The RADAR data are filtered and released as sparse point clouds with Doppler information. Data are annotated with 3D bounding boxes, classified between 23 classes, and provided in both Cartesian coordinates and camera domain. The **CRUW** dataset [Wang *et al.* 2021b] proposes 3.5 hours of recorded sequences in multiple scenarios (parking, campus road, city street, highway). The authors used a camera and a LD RADAR but they released only RA views. The provided annotations are single point object locations in the RA map with three classes. The recently released RADDet dataset [Zhang *et al.* 2021a] contains around 10,158 frames recorded in urban scenes with a stationary car mounted with stereo cameras and an LD RADAR. The dataset contains RAD tensors, RA and RD views; it does not contain RADAR point clouds. The annotations are 2D bounding boxes, classified in 6 classes, for each RADAR view. Additional details on the **RADDet** dataset and the annotation pipeline proposed by the authors are provided in

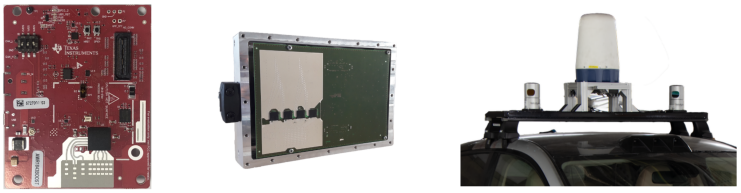


Figure 3.1: **Examples of Low-Definition, High-Definition and Scanning RADARs**. (Left) Low-Definition RADAR used in the CARRADA dataset (see Section 4.3), (middle) High-Definition RADAR used in the RADIal dataset (see Chapitre 6) and (right) Scanning RADAR on the roof of a car used in the Oxford RADAR RobotCar dataset [Barnes *et al.* 2020].

Section 5.1.4.1. Traditional RADARs have a poor angular resolution and multiple sensors are required to have a perception of the scene all around the car. This second limitation is overcome by scanning RADARs.

3.2.2 Scanning RADAR

Transmitter antennas of scanning RADARs generate signals successively to obtain a 360° Field-of-View (FoV) of the scene. The **Oxford RADAR RobotCar** dataset [Barnes *et al.* 2020] contains 280 km of recorded urban scenes in the city of Oxford at different periods of the year. The car is moving and mounted with three cameras, two 32-beam LiDARs and a 360° scanning RADAR. The provided RADAR representations are RA maps without annotation. The **RADIATE** dataset [Sheeny *et al.* 2021] is composed of 3 hours of RADAR sequences in diverse types of scenes (parking, urban, motorway and suburban) with various weather conditions. The moving car is mounted with a camera, a 32-beam LiDAR and a 360° scanning RADAR. The provided RA maps are annotated with 2D bounding boxes classified between eight classes. The **MulRan** dataset [Kim *et al.* 2020a] has a total of 41.2 km of recorded sequences in urban scenes within different cities at the month-level temporal gap. The authors used a 64-beam LiDAR and a 360° scanning RADAR to record RA representations without annotation on the RADAR data. The 360° scanning RADARs provide a high perception around the car with their large FoV, however their angular resolution is limited as traditional RADARs and they do not provide Doppler information. HD RADAR provides both improved angular resolution and Doppler information.

3.2.3 High-definition RADAR

Recent HD RADARs reach an azimuth angular resolution below the degree using large arrays of virtual antennas[3]. The **Astyx** dataset [Meyer & Kuschk 2019b] is composed of only 546 frames without temporal information. The authors used a HD RADAR coupled with a camera and a 16-beam LiDAR. The dataset provides a dense point

[3]HD RADARs contains hundreds of virtual antennas against tens for traditional or LD RADARs.

cloud including the Doppler information with 3D bounding box annotations, classified in 7 classes, in both Cartesian coordinates and image domain. The **Zendar** dataset [Mostajabi *et al.* 2020] is composed of 7.2 minutes of recorded sequences in urban scenes with RD, RA representations and RADAR point clouds. The scenes are recorded with a HD RADAR, a camera and a 16-beam LiDAR. The annotations are provided with 2D bounding boxes considering a single category in Cartesian coordinates. The **RadarScenes** dataset [Schumann *et al.* 2021] provides four hours for urban scene sequences with diverse weather conditions, traffic density and road classes (highway, city). The provided data are point clouds recorded from an HD RADAR and camera images. The camera images are annotated with semantic segmentation masks and the RADAR data with point-wise annotations. The objects are distinguished between 11 classes. None of these datasets provides ADC or RAD tensor data from a HD RADAR. HD RADAR datasets usually propose RADAR point clouds processed from the raw data. Unfortunately, the pre-processing pipeline looses information and requires a high computational cost.

3.2.4 Our proposals

In this thesis, we introduce two datasets to overcome the lack of annotated raw RADAR data for both LD and HD RADAR sensors. In Section 4.3, we propose the **CARRADA** dataset, the only dataset providing camera images with synchronised RAD tensors and their corresponding views annotated with bounding boxes and semantic segmentation labels. A semi-automatic annotation tool generating the annotations in the RADAR sequences is also presented. In Chapter 6, our collaborative work proposes the unique **RADIal** dataset composed of raw data recorded with HD RADAR together with camera and LiDAR in various driving environments, filling a gap in existing automotive RADAR datasets. Table 3.1 summarizes the characteristics of the publicly-available driving datasets with RADAR. The presented open source RADAR datasets have opened up research on scene understanding for automotive applications. The following sections will detail the related work on object detection, semantic segmentation and sensor fusion using RADAR data.

3.3 RADAR object detection

Classification of RADAR data for automotive applications has been little explored, *e.g.* for vehicle classification [Capobianco *et al.* 2017]. RADAR point clouds have been exploited for person identification with the temporal dimension using PointNet [Qi *et al.* 2017a] with an attention module [Cheng & Liu 2021] or with causal dilated temporal convolution for additional tracking [Pegoraro & Rossi 2021]. Classification is a restrictive task not suited to scene understanding, requiring to detect multiple objects in the environment of the car. The object detection task has been popular for the past few years to better exploit RADAR representations and improve the comprehension of the environment of a car.

3.3.1 Range-Angle-Doppler tensor

The entire Range-Angle-Doppler tensor has been explored to detect objects in the RA view corresponding to polar coordinates. In their work, [Major *et al.* 2019] proposed to aggregate the views of the RAD tensor by pair of axes, process them with individual CNN branches and stack their features as the input of a single decoder exploiting the temporal dimension to detect object in RA. The RAD tensor has also been aggregated in 2D views in the work of [Gao *et al.* 2020], each one being processed by a dedicated 3D autoencoder including the temporal information. The produced feature maps are fused and processed with an inception module for single point object location in the RA view. The authors also proposed data augmentation methods specialized for RA views. Additional details on the RAMP-CNN architecture proposed by [Gao *et al.* 2020] are provided in Section 5.1.2.2. In their work, [Palffy *et al.* 2020] localized an object in RA to crop the RAD tensor and classify the sub-RAD tensor with the Doppler information into a road user class. Recent work proposed to extract features from the RAD tensor with a ResNet backbone, considering the third dimension as channels [Zhang *et al.* 2021a]. Feature maps are then used to detect objects with two independent YOLO detection heads, one for 3D coordinates in the RAD tensor, the other for 2D coordinates in a Cartesian RA map. The RAD tensor is a cumbersome RADAR representation usually processed with a sub-representation to benefit from both the location and the Doppler of the reflectors. However, the entire tensor is not always available, therefore most of the previous related works are single view approaches.

3.3.2 Range-Angle or Range-Doppler view

Single view approaches consider either the Range-Angle or the Range-Doppler view to detect objects in one of them. Information related to the reflectors is reduced because it does not process the location and the Doppler information of the reflectors simultaneously. Considering RA view only helps to detect objects in a 2D space but it excludes the Doppler. The RA views have been used in a two-stage approach [Gao *et al.* 2019], a clustering localizes objects' signatures and a VGG architecture classifies them. Objects have been detected in RA views with single-stage approaches comparing the YOLO and SSD architectures [Stroescu *et al.* 2020]. Objects have been detected and classified in RA views of indoor scenes using a 300Ghz RADAR with an end-to-end CNN [Sheeny *et al.* 2020]. Objects have been detected in Cartesian RA with an architecture similar to SSD while minimizing the aleatoric uncertainty of the model, *i.e.* the variance of its predictions [Dong *et al.* 2020].

The recent CRUW dataset [Wang *et al.* 2021b] has opened up research on deep learning architectures for object location (single point detection) in RA view. The RODNet architecture proposed by [Wang *et al.* 2021a] processes RA with 3D convolutions and inception modules to localize objects with a single point considering a confidence map. Objects have been localised with a teacher-student method based on RODNet while improving spatio-temporal features with densely connected residual blocks based on atrous convolutions in both spatial and temporal dimensions [Hsu *et al.* 2021]. Multiple variants of the RODNet architectures have been considered [Sun *et al.* 2021] including temporal inception modules and fused their prediction to localize objects in RA. A multi-scaled U-Net architecture has

also been explored for this task [Ju *et al.* 2021] while proposing an inception module with 3D factorized convolutions. In their work, [Zheng *et al.* 2021] introduced a neural network with two branches, one recognizing urban scenes and the other localizing objects in RA. The authors also introduced SceneMix, a set of data augmentation methods for RA views specialized by type of scenes. More recently, [Azam *et al.* 2021] proposed to detect objects in a Bird's Eye View of RA by considering RADAR representation in grey scale transformed in RGB, LAB and LUV as the input of a transformer network using multi-head attention layers.

A few works have performed object detection in RD view providing the velocity information of the objects. However their position is ambiguous, only their distance can be deduced. Objects have been detected in RD using a YOLO architecture while defining anchor priors with the objects' signature in the grid of the representation [Pérez *et al.* 2019]. The U-Net architecture has also been considered by [Ng *et al.* 2020] to process simulated RD with complex values. In their work, [Lee, Wei-Yu *et al.* 2021] suggested a self-supervised method for feature alignment using self-weighted masks and a spatio-temporal consistency loss exploiting both annotated and non-annotated frames. Their method is applicable to both RA and RD views.

Other methods have proposed to process the RD view to deduce information about the location of the objects in the RA. A two-stage approach has been proposed to detect an object in the RD and then to estimate its angle [Brodeski *et al.* 2019]. Complex RDs have been processed with U-Net with complex convolutions to predict bounding boxes in RA [Stephan *et al.* 2021]. A recent work of [Meyer *et al.* 2021] exploited a tensor of RD view, without the third FFT on the antenna axis, to create a graph which is processed with graph convolutions and a ResNet Feature Pyramid Network (FPN) backbone for 3D bounding box detection in RA. These methods try to localise objects without the entire RADAR information. Advanced neural network architectures have been proposed but they do not exploit both the location and the Doppler information of the reflectors. The point cloud RADAR is a lighter representation than the RAD tensor with less information about each object but it carries both their location and Doppler.

3.3.3 RADAR point cloud

The RADAR representation in Cartesian point cloud contains both the Doppler (excepting for scanning RADAR) and the RCS information. The point cloud is either sparse or dense depending on the RADAR sensor (LD or HD). In their work, [Danzer *et al.* 2019] processed a 4D RADAR point cloud (Cartesian coordinates, Doppler and RCS) with modified Frustum PointNets [Qi *et al.* 2018] generating patch proposals, classifying each patch, segmenting the points in each patch and regressing the 2D bounding box coordinates. The work of [Scheiner *et al.* 2020] used RADAR point clouds to detect, classify and track masked objects considering reflected waves received by the RADAR with a specific AutoEncoder (AE) architecture. The PointNet++ [Qi *et al.* 2017b] architecture has been explored by [Svenningsson *et al.* 2021] to process RADAR point clouds with a 3D convolutional FPN and two output branches, one detecting objects and the other performing tracking. The work of [Svenningsson *et al.* 2021] proposed to consider

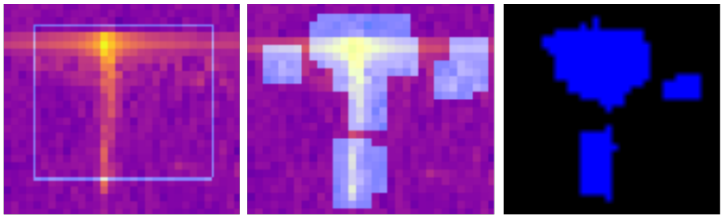


Figure 3.2: **Example of the annotation of a 'Car' signature in Range-Doppler**. The annotations of a 'Car' signature in a cropped Range-Doppler view are compared between (a) bounding box and (b) semantic segmentation. The bounding box annotation includes speckle noise while the segmentation mask is well suited to the object signature. Our proposed TMVA-Net neural network architecture detailed in Section 5.1.2.3 successfully segments the object signature (c).

PointNet++ features as a graph and apply graph convolutions to generate contextual embeddings used to detect and classify objects in a BEV representation of the sparse point cloud. A dense point cloud of a HD RADAR has been processed for 3D object detection using a self-attention mechanism to learn global pillar[4] features used as input of a 3D CNN and an RPN head [Xu *et al.* 2021]. These methods considered point clouds from either LD or HD RADAR. However, their raw signals are processed to create the point cloud and objects may be missed as explained in Section 2.2.

The object bounding box detection task has been widely explored but it is not well suited to RADAR data. Objects' signatures have non-compact shapes, they are extended on an axis of the representation and they have various sizes as illustrated in Figure 3.2. Bounding boxes include noise and sometimes shapes from different categories. The next section will detail the related work on RADAR semantic segmentation to overcome these issues.

3.4 RADAR semantic segmentation

The characteristics of the object signature in RADAR representations make semantic segmentation well suited for scene understanding (see Figure 3.2). The shapes are non-compact due to the surface of the objects which reflecting the signals. Moreover, objects can be close to each other and partially mixed due to the sensor resolution. The pixel-wise segmentation of the representations takes these problems into account with more accurate predictions. RADAR semantic segmentation has not been extensively explored due to the lack of annotated data. The CARRADA dataset opened up research in this field thanks to the semi-automatic pipeline generating annotations, including semantic segmentation, proposed in Section 4.3.2. Nonetheless, to the best of our knowledge, there is no previous

[4]Point clouds are sometimes transformed into a pillar representation: each point with a 3D Cartesian position has its height axis extended with a fixed value.

work on semantic segmentation using the entire RAD tensor as our method proposed in Section 5.1. The following sections detail applications of RADAR semantic segmentation on single RADAR views or on RADAR point clouds.

3.4.1 Range-Angle view

To the best of our knowledge, there is no previous work on RADAR semantic segmentation on RD views; this section thus focuses on RA views only. Occupancy grid segmentation is a popular area of research using RADAR data. The work of [Lombacher *et al.* 2017] proposed a segmentation of Cartesian RA maps in BEV to distinguish occupied (cars or others) and free pixels using a 6-layer CNN architecture. The segmentation of Cartesian Bird's Eye View RA map has been extended to classify the pixels between static (occupied or free) and dynamic road users, using an AutoEncoder architecture with up-convolutions while predicting the motion of dynamic objects [Hoermann *et al.* 2018]. In their work, [Prophet *et al.* 2019] generated occupancy grids from a dense HD RADAR point cloud and segmented them between four classes using FCN-8s [Long *et al.* 2015], U-Net [Ronneberger *et al.* 2015] and SegNet [Badrinarayanan *et al.* 2017]. The DeepLabv3+ [Chen *et al.* 2018b], FCN and a lightweight version of FCN have been compared for binary occupancy segmentation of RA views in both Cartesian and polar coordinates [Nowruzi *et al.* 2020]. Occupancy grids have also been generated using LiDAR and sparse RADAR point clouds [Sless *et al.* 2019] and segmented using a simple CNN architecture similar to U-Net. HD RADAR point clouds have been used to create 3D occupancy grid [Prophet *et al.* 2020], projected in 2D BEV for occupancy segmentation using Deeplabv3+ and SegNet. The work of [Kaul *et al.* 2020] proposed RSS-Net, the first architecture for RADAR semantic segmentation trained on Cartesian Bird's Eye View RA maps with seven categories. They also proposed a method to transfer the annotations from the LiDAR sensor to the RADAR representation. The RSS-Net architecture is similar to DeepLabv3+ with an ASPP module, it is trained with a custom weighted cross entropy loss. Additional details on the RSS-Net architecture are provided in Section 5.1.2.2.

3.4.2 RADAR point cloud

The point cloud segmentation task consists in classifying each point in a category. Point clouds are usually represented in BEV and Cartesian coordinates due to the low elevation resolution of the RADAR sensor. The PointNet++ architecture [Qi *et al.* 2017b] has been used to segment dense HD RADAR point clouds [Feng *et al.* 2019]. In their work, [Schumann *et al.* 2018] segmented 2D RADAR point clouds using PointNet++ with additional multi-scale grouping and feature propagation modules. The PointNet++ has also been extended to segment RADAR point clouds with a mean shift [Comaniciu & Meer 2002] feature extractor learning local dependencies and a feature fusion module based on attention mechanism [Cennamo *et al.* 2020]. The work of [Cheng *et al.* 2021a] proposed to use the coordinates of the points and the RD view as inputs of a segmentation network similar to U-Net in order to classify each point as occupied or free. RADAR point clouds in BEV have been segmented for anomaly detection

[Griebel *et al.* 2021], as ghost or multi-reflection, using PointNet++. A 4D RADAR point cloud from multiple RADARs has been explored by [Schumann *et al.* 2021], combining Cartesian coordinates, Doppler and RCS, and processed with three PointNet++ models and a recurrent point classifier for instance point segmentation. Instance segmentation has also been explored using HD RADAR with PointNet++ and an additional clustering method processing the semantic information of the point clouds [Liu *et al.* 2021a]. Recently, the kernel point convolution (KPConv) layer and a variant of the LSTM cell for temporal dimension processing have been proposed by [Nobis *et al.* 2021a] for semantic segmentation of sparse RADAR point clouds.

RADAR semantic segmentation is an interesting approach for scene understanding as detailed in Section 5.1. However, many situations are difficult to understand using RADAR alone because of its low angular resolution and the difficulties of human interpretation. The following sections will review existing works on sensor fusion using RADAR with camera and, or, LiDAR providing improvements in many tasks.

3.5 Sensor fusion

3.5.1 RADAR and camera fusion

The camera is a passive sensor that provides a dense and comprehensive representation. It has been usually preferred to be fused with the RADAR sensor as they are complementary: they respectively provide dense semantic information and the location and Doppler of the objects. Camera and RADAR fusion methods commonly use sparse RADAR point clouds. The following related works are differentiated among three fusion approaches: early (at the data level, the input of the network), mid-level (at any feature level of the network) and late (at the prediction level, or last layer of the network). In the presented methods, the RADAR data have been used to improve performances in image-based tasks.

Early fusion. Early fusion methods project RADAR point clouds in the image using the calibration of the camera. The precursor work of [Garcia *et al.* 2012] was not a machine learning method, but the authors projected a sparse RADAR point cloud in the camera image to perform block matching and track objects according to their Doppler information. The work of [Kowol *et al.* 2021] expended the RADAR points on the entire height of the image, w.r.t. the poor RADAR elevation resolution, and used it as input of a YOLO model to perform 2D object detection. Image depth estimation has been explore by [Long *et al.* 2021b] with a two-stage approach. The authors trained a network to associate RADAR points to the objects in the image with a confidence area and a second model predicts the depth map with the fused camera and RADAR points joined with the confidence areas. In their work, [Long *et al.* 2021a] explored full velocity estimation of sparse RADAR points by mapping a RADAR point to an object in an image, as in the previous method, and deducing the closed form of the velocity vector of the point using the optical flow of the object. The RADAR point coordinates have been projected in the image plan and fused with the image to learn an association between feature vectors of the

RADAR points and the corresponding object bounding boxes [Dong *et al.* 2021]. RADAR point clouds have been aligned with detected objects using a two-path Faster R-CNN for 2D object detection [Liu *et al.* 2021c].

Mid-level fusion. Mid-level fusion methods process the image and the RADAR data with independent neural network backbones and merge their feature maps in a common latent space. In the work of [Chadwick *et al.* 2019], RADAR points have been projected in the image plan, two ResNet branches process the camera image and the RADAR point cloud image producing feature maps which are fused and used as input of detection heads similarly to the SSD architecture improving long range detection. In their work, [Nobis *et al.* 2019] used a similar point cloud processing but the two CNN branches are linked with intermediate connections at each stage, a final FPN detects 2D objects while an additional module focuses the weights on a certain sensor. In the same manner, [Yadav *et al.* 2020] fused the features of the two branches in a mid-level stage of an FPN for 2D object detection. Feature maps have been fused by considering an SSD architecture with RADAR convolutional attention modules balancing the sensors and improving detection of small objects [Bai *et al.* 2020]. The work of [Meyer & Kuschk 2019a] used HD RADAR point clouds with the camera image as the input of an AVOD architecture [Ku *et al.* 2018] producing representations of each modality with 3D anchors and 3D proposals from the RPN. The authors showed better performances than the same pipeline using LiDAR point cloud combined with the camera image. In their work, [Nabati & Qi 2020] detected 3D objects by processing RADAR point clouds in the image (early fusion) with convolutions to detect objects and an additional refinement block takes the RADAR point cloud to refine proposal coordinates. In the work of [Niesen & Unnikrishnan 2020], the entire RA view has been processed with the image considering individual branches. Their feature maps have been fused and up-sampled with a decoder for 3D depth reconstruction. The RADAR point cloud has also been projected in the image plane with Gaussian kernels creating a range density map [Cheng *et al.* 2021b] and processed as an image. The authors proposed a vision-RADAR fusion based on self-attention and global-attention layers used as input of an FPN for 2D object detection. This method showed improvements in small and long range object detection. HD RADAR point clouds have been considered in a BEV and front view representations, processed with individual backbones [Cui *et al.* 2021]. Their feature maps are fused for 3D object detection and tracking is performed using a Kalman filter. In the work of [Zhang *et al.* 2021b], RADAR point clouds have also been used to predict BEV occupancy grid. The authors have projected images from multiple cameras in a BEV representation with a specific neural network and its feature maps from both branches have been fused for 2D object detection. Traffic flow has been estimated in the work of [Jin *et al.* 2021], *i.e.* consisting in 3D bounding boxes, speed and traffic density estimation. The authors used a point pillar expansion on RADAR point clouds and fused them with image features. They also used a CycleGAN [Zhu *et al.* 2017] generating image data with different lighting conditions showing the limitations of image-based methods. Depth estimation has been explored using a self-supervised learning scheme by expanding RADAR point cloud for weak supervision and image reconstruction [Gasperini *et al.* 2021].

Late fusion. These methods consist in fusing feature maps generated by individual backbones at the very end of the pipeline, before the predictions. In their work, [Lekic & Babic 2019] used Cartesian RA views as input of conditional Generative Adversarial Networks (cGANs) [Isola *et al.* 2017]. Their outputs are fused lately to generate free space segmentation in camera images from RADAR occupancy grid representation and segmented images. RADAR point clouds have been used to generate 2D proposals for 2D object detection (mid-level fusion), each modality having its own network. Their features are finally fused before the bounding box predictions [Shuai *et al.* 2021]. The work of [Kim *et al.* 2020c] used individual backbones to process images and RADAR voxel grids creating features maps which are fused and used as input of a 3D RPN (mid-level fusion). The feature maps are then transmitted to the detection head with a gated RoI fusion module. They showed better performances in middle and long range detection than with LiDAR point clouds. Similarly, [Kim *et al.* 2020b] generated features maps from images and RA with independent backbones as input of a 3D RPN (mid-level fusion). The Cartesian RA are also used at the detection head level. In their work, [Kuang *et al.* 2020] fused feature maps from each bounding box and the RADAR points to create an affinity matrix with their features and proposed a dynamic coordinates alignment method to refined the detection predictions. In the work of [Li & Xie 2020], the RADAR points have been projected in the camera plan and expanded with pillars and circles. Then, the authors used a YOLOv3 [Redmon & Farhadi 2018] with an FPN attention to fuse the RADAR image with the feature maps at different scales. RADAR point clouds have been projected in the image plan and processed with convolutions in the work of [Hussain *et al.* 2021]. They processed the image in parallel with an FPN and fused their feature maps which are up-sampled for depth map estimation. In their work, [Lo & Vandewalle 2021] proposed a similar approach projecting the RADAR point clouds in the image plan to create a RADAR depth map; then, separated backbones generate representations of the modalities which are fused before an ordinal regression [Fu *et al.* 2018] for depth map estimation. A two-stage method for depth estimation has also been proposed by [Lin *et al.* 2020]. A first CNN generates a coarse depth map to filter RADAR outliers using images with project RADAR points. In the second stage, the coarse map, the original image and the filtered image with RADAR points are processed with a second network predicting the dense depth map. In the work of [Nabati & Qi 2021], expanded RADAR points in pillars have been associated to object bounding boxes. Then, the modalities are processed by individual backbones and their feature maps are stacked for 3D bounding box and velocity estimation. Fusing camera and RADAR representations have increased performances in vision-based tasks. However the fusion is not easy since the modalities are not recorded in the same space. The following section will detail methods fusing RADAR and LiDAR sensors.

3.5.2 RADAR and LiDAR fusion

The LiDAR sensor provides dense 3D point clouds of a scene in polar coordinates. However, it does not provide the object velocity and its point cloud is dense only at short range. These limitations are overcome by using a RADAR sensor recording Doppler information with a longer maximum range capability. RADAR and LiDAR fusion has not been deeply

explored due to the lack of open source datasets. The related work is generally based on early fusion since both representation can be projected in a Cartesian space.

The work of [Weston *et al.* 2019] proposed an inverse sensor model (ISM) converting a Cartesian BEV of the Range-Angle RADAR view into an occupancy probability grid based on its uncertainty. The model is trained with self-supervised learning based on an occupancy grid created from the LiDAR point cloud.

In their work, [Shah *et al.* 2020] suggested a pipeline processing a HD map, LiDAR and RADAR point clouds in BEV representations with individual backbones producing feature maps which are fused for 2D object detection and trajectory prediction. The RADAR backbone is composed of a spatial network extracting sparse local information using parametric continuous convolutions, and a temporal network using MLP to combine the spatial features in the temporal axis acting as an attention layer.

A voxel-based early fusion for better long range detection has also been explored by [Yang *et al.* 2020] and combined with an attention-based late fusion to perform 3D object detection and trajectory estimation using two detection heads from a PnPNet architecture [Liang *et al.* 2020].

RADAR Bird's Eye View RA has been fused with LiDAR at the early stage of the network to predict RADAR occupancy grid with a U-Net architecture using LiDAR supervision [Kung *et al.* 2021]. The authors proposed a sliding window approach to train the network at short range and demonstrated generalization outside the LiDAR sensing range using the RADAR only to predict the occupancy grid.

The work of [Qian *et al.* 2021] processed RADAR BEV RA and LiDAR BEV map, corrupted with simulated fog, with independent U-Net backbones. An RPN extracts proposals from their fused feature maps (mid-level fusion). Each modality has an RoI pooler producing a feature vector from each proposal which is fused with self-attention and cross-attention (late fusion) for 2D object detection.

Realistic Cartesian RADAR scenes have been generated by improving the elevation of RADAR points using an adversarial approach with simulated elevation and real LiDAR elevation considered as ground truth [Weston *et al.* 2021].

Methods exploiting combined RADAR and LiDAR data without deep learning have recently emerged. In the work of [Farag 2021] and [Liu *et al.* 2021b], authors proposed to detect objects with clustering in LiDAR and RADAR representations independently. Their features are fused and the objects are tracked using a custom Kalman filter algorithm while estimating their velocity individually. RADAR and LiDAR data have been combined to detect surfaces in adverse weather conditions highlighting robustness against fog [Wallace *et al.* 2021].

Fusion of RADAR and LiDAR data in Cartesian coordinates benefits from the density of the LiDAR and the long range with Doppler of the RADAR. It is still at the early stage, fusion methods and adapted neural network architectures will be explored in the future. In Section 5.2, we will introduce a fusion method propagating RADAR information through the LiDAR point cloud. The following section will detail methods for RADAR, camera and LiDAR sensor fusion.

3.5.3 RADAR, camera and LiDAR fusion

Combining RADAR, camera and LiDAR requires to project the data or their representations in a common space leading to complicated pipelines to perform end-to-end learning.

The work of [Bijelic *et al.* 2020] projected LiDAR and RADAR point clouds in a range view, *i.e.* the image view, and extended the RADAR points on the entire height of the image. The entropy of each modality is estimated and used as input of the CNN backbone corresponding to its modality. The entire pipeline is similar to SSD [Liu *et al.* 2016] with additional connections linking each backbone and heads for 2D detection in the camera (details about the SSD architecture are provided in Section 2.6.2).

The architecture of [Shah *et al.* 2020] introduced in the previous section has been modified by including camera images and feature gaits simulating sensor dropout to perform 3D object detection and trajectory estimation [Mohta *et al.* 2020]. It uses a MultiXNet [Djuric *et al.* 2021] feature extractor specialized in multi-modality processing. The authors showed that their method reduces sensor over-fitting while improving generalization when a sensor is missing.

Partial optical flow has been explored by [Grimm *et al.* 2020] considering camera to localize objects, LiDAR as a label for depth estimation and RD RADAR view to estimate velocity vectors pixel-wise. They proposed to learn a module warping the segmented RD map into the image domain and to use the depth map from the LiDAR point cloud to supervise the training.

In their work, [Nobis *et al.* 2021b] projected image pixels in the LiDAR point cloud and fused RADAR and LiDAR point clouds in voxels for 2D object detection. Features are extracted using a VoxelNet [Zhou & Tuzel 2018], processed with 3D sparse sub-manifold convolutions [Graham *et al.* 2018] and projected in a dense BEV representation to apply standard convolutions for the detection heads.

Similarly, [Wang *et al.* 2020a] associated an image pixel to each LiDAR point with the temporal dimension creating a first 7D point cloud, and grouped RADAR features to obtain a 8D RADAR point cloud. Both point clouds are processed separately with the frustum PointNets framework [Qi *et al.* 2018] and their features are lately used for 3D object detection and velocity estimation.

RADAR, camera and LiDAR fusion is still at its early stage, there is neither clear common latent representation to fuse all the representations nor relevant backbone extracting features from the three sensors simultaneously.

3.6 Conclusions

In recent years, deep learning algorithms applied to autonomous driving have left out the RADAR sensor for scene understanding. The nuScenes dataset has been the first to propose sparse RADAR point clouds without annotation. Since then, datasets were released including various scenes, annotations and sensors (see Table 3.1). The Chapter 4 will discuss methods to tackle the lack of RADAR data for scene understanding. In particular, Section 4.3 will detail our proposed CARRADA dataset and a semi-automatic pipeline to generate its annotations. Result of a recent collaboration, our RADIal dataset including raw

HD RADAR data is presented in Chapter 6. The two proposed datasets are unique in their kind; they are publicly available to the machine learning community to support research in scene understanding.

Deep learning for RADAR scene understanding has been opened up with the release of large scale datasets. RADAR semantic segmentation has not been widely explored but it is the most suited task regarding the objects' signature in the RADAR representation. Moreover, exploiting raw RADAR data is important since the pre-processing steps reduce information and miss small object signatures. The Range-Angle-Doppler tensor is noisy and cumbersome and thus should be aggregated in views. To the best of our knowledge, there is no related work on RAD tensor segmentation for scene understanding. In Section 5.1, we propose the first approach segmenting multiple views of the RAD tensor simultaneously outperforming competitive methods while requiring significantly less parameters.

To the best of our knowledge, there is no previous work on end-to-end object detection that is capable to scale with raw HD RADAR data. Moreover, there is no previous work either on free driving space segmentation[5] or semantic segmentation using only RD views of HD RADAR signals. In addition, there is no existing multi-task model that performs both RADAR object detection and semantic segmentation simultaneously. In Section 6, we propose a deep neural network architecture learning the costly pre-processing steps and performing multi-task learning using raw HD RADAR data: 2D object detection and free space segmentation simultaneously.

Sensor fusion including RADAR has been recently explored. It has shown improvements in long range detection, small object detection and global performances in adverse weather conditions. LiDAR and RADAR fusion has not been extensively investigated although the two type of sensors have complementary properties and they can be both represented in a Cartesian space. In Section 5.2, we propose a preliminary sensor fusion approach aiming to propagate RADAR information through a dense LiDAR point cloud.

[5]The free driving space segmentation task consists to locate pixel-wise the available space that can be driven. Further details are provided in Chapter 6.

CHAPTER 4

Proposed automotive RADAR datasets

Contents

4.1 RADAR simulation 58
4.1.1 Parameters and properties 58
4.1.2 RadarSim dataset 59
4.1.3 Experiments and results 61
4.1.4 Discussions 62
4.2 RADAR data generation 63
4.2.1 Dataset 63
4.2.2 Range-Doppler representation 64
4.2.3 Methods and Experiments 66
4.2.4 Discussions 70
4.3 CARRADA dataset 70
4.3.1 Dataset 71
4.3.2 Pipeline for annotation generation 72
4.3.3 Semantic segmentation baseline 79
4.3.4 Discussions 81
4.3.5 Conclusions 81
4.4 Conclusions 82

RADAR sensors generate electromagnetic wave signals that are not affected by weather conditions or darkness. These sensors inform not only about the 3D position of other objects, as LiDAR, but also about their relative speed (radial velocity). However, in comparison to other sensory data, RADAR signals are difficult to interpret, very noisy and cumbersome while having a low angular resolution. RADAR has been left behind LiDARs and cameras for these reasons and thus, has not been integrated in the data recording campaigns of large automotive datasets.

As detailed in Section 3.2, tackling the lack of open source datasets has been a challenge in recent years. In this chapter, a simple RADAR simulation is proposed in Section 4.1, a method to generate RADAR data is detailed in Section 4.2; and a novel dataset with synchronised camera and RADAR data and a semi-automatic algorithm generating its annotations is described in Section 4.3.

Section 4.3 presents a work mainly inspired from our article published at the International Conference on Pattern Recognition (ICPR) [Ouaknine *et al.* 2020].

4.1 RADAR simulation

At the beginning of 2019, there was no open source RADAR dataset for automotive scene understanding. A costly and time consuming solution is to record and annotate a dataset. A second one is to consider a simple simulation of the behavior of a RADAR sensor. The simulation has two major benefits: it is an unlimited source of data and fine-grained annotations are available without any cost. In this first approach, we propose a simple simulation of the RADAR sensor based on geometric considerations. Starting from a RADAR position and a moving object and knowing its size and trajectory, we can deduce its distance to the sensor and its relative velocity and thus predict its approximated signature in the RD data. We have created a simulated dataset named RadarSim to train a classifier to distinguish RD sequences between four categories of objects. Object classification experiments have been conducted using deep neural network architectures.

4.1.1 Parameters and properties

The simulation starts from a class of objects (among 4 categories) and a straight line trajectory in a delimited 2D space. Each object is represented by a square whose size depends on its category. The real velocity and the angle of the object direction define a vector in polar coordinates which will determine the trajectory of the object. The category also defines the range of values for the trajectory parameters (real velocity and angle).

The object is moving in front of a RADAR, the sensor is considered as a single stationary point. The transmitted wave and its reflection on the object are not modeled. Instead, several points belonging to the reflector are selected and the simulation computes their relative velocity and distance w.r.t. the RADAR position.

Four categories are supported by the simulation: pedestrian, motorcycle / bicycle (same category), car and truck / bus (same category). An integer value is associated to each category, ordered in order of the size of the object. The category defines the size of the objects, the higher the larger. For each one of them, the size parameter is expressed in meters and is randomly drawn as $l \sim \mathcal{U}([l_{\text{min}}, l_{\text{max}}])$. The range of values depends on the category, see Table 4.1. Each object has a real velocity expressed in meters per second ($m \cdot s^{-1}$). Its value is randomly drawn as $v \sim \mathcal{U}([v_{\text{min}}, v_{\text{max}}])$, where the estimated range of real world velocities of the objects also reported in Table 4.1. The time interval between two frames is fixed to $\Delta t = 0.1$ second.

While the object is moving, a vector (angle and length) defines its direction. The angle of direction of the object is randomly drawn as $\alpha \sim \mathcal{U}([0, 2\pi])$ while its length depends on the velocity previously defined. Depending on this value, the object will start to move in one of the four regions of the space. These splits are made to ensure that the object will not cross the boundaries of the space. Only a few points belonging to the object are estimated to create the signature. For each one of them, the intensity (in dB) of the reflection is

Category	Size l (m)	Velocity v ($m \cdot s^{-1}$)	Intensity I (dB)
0: Pedestrian	$[0.1; 0.5]$	$[1.0; 3.0]$	$[50.0; 65.5]$
1: Motorcycle / Bicycle	$[0.5; 2.0]$	$[3.0; 11.0]$	$[50.0; 80.0]$
2: Car	$[2.0; 4.0]$	$[4.0; 14.0]$	$[65.5; 80.0]$
3: Truck / Bus	$[4.0; 10.0]$	$[4.0; 14.0]$	$[65.5; 80.0]$

Table 4.1: **Range of values for the simulated RADAR points properties**. An object is defined by its category, size (in meter), its velocity (in meter per second) and the estimated intensity of its reflected signal (in decibel).

estimated according to real observations and drawn as $I \sim \mathcal{U}([I_{\text{min}}, I_{\text{max}}])$. The ranges of intensity values are defined in Table 4.1.

The RD representation contains a high level of noise divided into three forms when considering averaged and log-transformed data (see Section 4.2.2): speckle noise, uniform noise and zero-Doppler noise. The logarithmic transformation aims to transform the speckle from a multiplicative to an additive noise as $I = u \times s$, where I is the intensity, u the underlined reflectivity and s the speckle as detailed in Section 2.2.2. The simulation considers log-intensity values and thus includes the speckle noise as a global additive noise. We have estimated the Fisher-Tippett distribution of the background (combining the reflectivity of the background and the speckle) from real log-transformed data [Goodman 2007]. It can be approximated by a Gaussian distribution as illustrated in Figure 4.3 since the data have been multi-looked. Regarding our estimations, the simulation will use a random variable $s \sim \mathcal{N}(35.473, \sqrt{0.244})$ to generate the background of the representation, including the speckle noise. We also introduce a uniform noise to mimic reflections of false alarms that can occur in recordings (ghost or multi-path reflections, ambiguity, and so on). It will activate each bin according to a random variable defined with a Bernoulli distribution $r \sim \text{Bern}(0.01)$. Finally, the zero-Doppler noise will contain all the reflections which have a zero Doppler value (stationary objects) at each range bin. This phenomena is modeled on the zero Doppler bins with a random variable defined with a Bernoulli distribution defined as $z = \text{Bern}(p)$ at the 0 range bin, where the probability parameter p decreases from 1 by a factor 0.005 for each range bin. For a single bin of the representation, the simulated background including the noise is thus a random variable written $n = s + r + z \cdot \mathbb{1}_{\text{Doppler}=0}$ which is added to the simulated log-intensity. The following section will present the entire pipeline of simulation and the RadarSim dataset.

4.1.2 RadarSim dataset

The dataset simulation starts by defining the object: randomly drawing its category, real velocity, direction and its starting point in the space. A sequence of twenty RD frames is generated (equivalent to two seconds of recordings). Each one of them will be a matrix of size 220×100. An example of a frame from a generated scene is shown in Figure 4.1. The following process is executed at each timestamp:

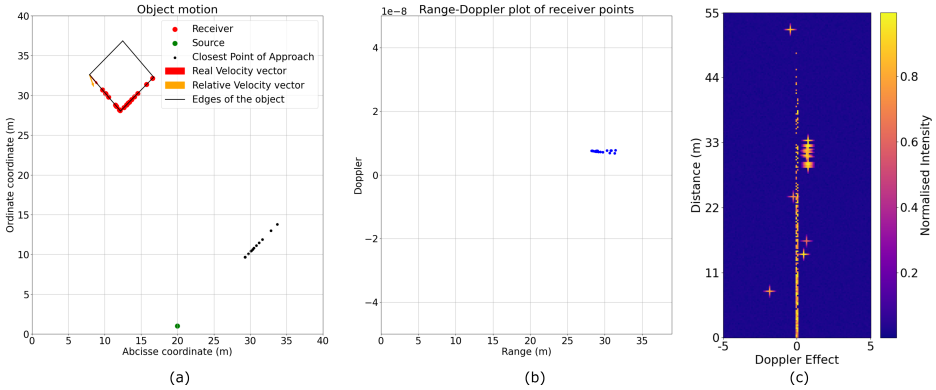


Figure 4.1: **Example of a Range-Doppler simulation for the category 'Car'**. (a) Simulation of the objects motion in a Cartesian space with a constant velocity vector and a fixed RADAR. The closest position of each sampled point w.r.t. radar position and the velocity of the object is noted the closest point of approach. (b) Range-Doppler points of the simulated reflections on the object. (c) Range-Doppler representation with the simulated object signature and additional noise phenomena.

- The range-Doppler matrix is created by discretizing the estimated values: the range is defined between 0 and 55 meters with a resolution of 0.25 meters per bin; the radial velocity (Doppler) is defined between -5 and 5 meters per second with a resolution of 0.1 per bin. The matrix is initialized with the estimated background and the randomly generated noise.

- Random points are selected on the visible edges of the moving object that are visible to the RADAR.

- The radial velocity component w.r.t. the RADAR is deduced from the real velocity vector for each point as well as their distance.

- For each point, an intensity value is randomly drawn depending on the category of the object. An additional *sinc* effect is considered to simulate the shape of the object reflection in the RADAR representation. The cardinal sinus function is defined as: $\mathrm{sinc}(x) = \frac{\sin(\pi x)}{\pi x}$.

The simulation process generating a sequence of RDs is detailed in Algorithm 1.

The generated RadarSim dataset contains 1000 sequences with 20 frames for each scene. Each scene lasts 2 seconds, which means that an RD is recorded every 0.1 second. The probability of randomly drawing a category is set to 0.25 to obtain a well-balanced data set. The dataset is split into training (60%), validation (20%) and test (20%). The classification task consists in categorizing sequences between four categories. Depending on the model used, it either classifies each frame of a sequence or directly classifies the entire sequence. Further details are provided in the following section.

Algorithm 1 Simulation of a sequence of Range-Doppler (RD)

Require: $T, e \in \mathbb{N}$, $(x_{\text{RADAR}}, y_{\text{RADAR}}) \in \mathbb{R}^2$, $d_{\text{max}}, r_{\text{max}} \in \mathbb{R}$

T the number of frames in the sequence, e the number of bins affected by the $sinc$ function, $(x_{\text{RADAR}}, y_{\text{RADAR}})$ the coordinates of the RADAR sensor, d_{max} and r_{max} the maximum Doppler and distance.

$p = 1$ # Initialize the parameter of the random variable z.

$c \in \{0, 1, 2, 3\}$ # Set the class of the object.

$x \sim \mathcal{U}([x_{\text{min}}, x_{\text{max}}])$ # Init. x coord. of the object in a bounded Cartesian space.

$y \sim \mathcal{U}([y_{\text{min}}, y_{\text{max}}])$ # Init. y coord. of the object in a bounded Cartesian space.

$l \sim \mathcal{U}([l_{\text{min}}, l_{\text{max}}])$ # Set l the size of the object depending on c.

$v \sim \mathcal{U}([v_{\text{min}}, v_{\text{max}}])$ # Set v the velocity of the object depending on c.

$\alpha \sim \mathcal{U}([0, \pi]$ # Set α the angle of arrival of the object.

$s \sim \mathcal{N}(35.473,\ \sqrt{0.244})$ # Random variable for the speckle noise.

$r \sim \text{Bern}\,(0.01)$ # Random variable for the false alarm reflection.

$z \sim \text{Bern}\,(p)$ # Random variable for the zero-Dopppler noise.

$n = s + r + z \cdot \mathbb{1}_{\text{Doppler}=0}$ # Global random variable for noise and background generation.

for $t = 0$ **to** $T - 1$ **do** # T the length of the sequence to be generated.

  $\mathbf{A}$: 220×100 matrix filled with n where p decreases by 0.005 for each range bin.

  K: random points on the visible edges of the objects + number of false reflections.

  for $k = 0$ **to** $K - 1$ **do** # K the reflected points.

    $I \sim \mathcal{U}([I_{\text{min}}, I_{\text{max}}])$ # Set I the intensity of the reflection depending on c.

    d: compute the radial velocity component of k

    $r = \sqrt{(x - x_{\text{RADAR}})^2 + (y - y_{\text{RADAR}})^2}$ # Compute the range of the point k

    $\mathbf{A}_{\lceil \frac{d_{\text{max}}}{d} \rceil, \lceil \frac{r_{\text{max}}}{r} \rceil} \leftarrow \mathbf{A}_{\lceil \frac{d_{\text{max}}}{d} \rceil, \lceil \frac{r_{\text{max}}}{r} \rceil} + I$ # Add the reflected point k to the RD.

    $\mathbf{A}_{\lceil \frac{d_{\text{max}}}{d} \rceil \pm e, \lceil \frac{r_{\text{max}}}{r} \rceil \pm e} \leftarrow \text{sinc}(\mathbf{A}_{\lceil \frac{d_{\text{max}}}{d} \rceil \pm e, \lceil \frac{r_{\text{max}}}{r} \rceil \pm e})$ # Apply sinc on e bins around the reflection.

    $x \leftarrow x + v_x \times \cos(\alpha)$ # Update x w.r.t. the velocity and orientation of the object.

    $y \leftarrow y + v_y \times \cos(\alpha)$ # Update y w.r.t. the velocity and orientation of the object.

4.1.3 Experiments and results

Each sequence of the RadarSim dataset represents the RD signature of an object belonging to one of the four categories. The objective of the experiments is to classify each RD matrix or each sequence into a category depending on the training approach. For this purpose, a

Architecture	Approach		Accuracy	
	Frame	Sequence	Validation	Test
ResNet-101 [He *et al.* 2016]	✓	✗	87.5	90.0
ResNet-101 [He *et al.* 2016]	✗	✓	83.0	85.0
ResNet-34 [He *et al.* 2016] + LSTM (3 cells)	✗	✓	84.0	84.5
3D Convolutions (2 layers + 1 FC)	✗	✓	63.0	59.0

Table 4.2: **Performances on the RadarSim-Val and -Test datasets**. Performances of the explored architectures on the validation and test splits of the RadarSim dataset.

classifier $f_\theta(.)$ parameterized by $\theta \in \mathbb{R}^d$ is trained. The complexity d of the model depends on the chosen neural network architecture. The parameters are updated with the gradient based method Adam [Kingma 2015] for a fixed learning rate value. The parameters θ of the function are initialized using the Xavier initialization method [Glorot & Bengio 2010]. The Cross-Entropy loss is optimized during training:

$$\min_{\theta \in \mathbb{R}^d} \sum_{i=0}^{S-1} \mathbf{y}_i \log(f_\theta^*(\mathbf{x}_i)), \tag{4.1}$$

where S is the number of sequences, $\mathbf{y}_i$ the labels of the RD sequence $\mathbf{x}_i$ and $f_\theta^*(.)$ the estimated model. Performances are evaluated on the validation and the test datasets using the accuracy metric defined as: $\frac{\text{TP}}{\text{TP+FN}}$, where TP is the number of True Positive predictions and FN the number of False Negative predictions.

The performances are evaluated at the sequence level. A few architectures do not take into account the time dimension and thus predict a class for each frame of the sequence: this is the frame-based approach. During the training of the second approach, the loss is computed using the predictions and ground-truths of each frame. While during validation and testing, the accuracy is computed after a vote for each sequence regarding the predictions over its frames. For a given sequence $\mathbf{x}$, its attributed category is voted as following:

$$\underset{k \in \{0,\dots,3\}}{\arg\max} \sum_{i=0}^{N-1} \mathbb{1}_{\{f_\theta^*(\mathbf{x}_i)=k\}}. \tag{4.2}$$

The temporal information is incorporated by concatenating the frames of a sequence in the channel dimension to directly classify the sequence. The sequence-based approach consists in taking into account the time dimension and classifying the sequence with a single label.

Each architecture has been explored with several parameters including its number of layers. The highest performances for each architecture are reported in Table 4.2 and discussed in the following section.

4.1.4 Discussions

Results presented in Section 4.1.3 show that the architectures used have succeeded in estimating the parameters of the simulation. The generated scenarios are not challenging

enough and it is easy for the system to overfit the training dataset since the distributions between the splits are similar. We noticed that a light backbone with LSTM cells leads to similar results as those of extremely deep neural networks. Moreover, a model using 3D convolutions with only two layers has reached almost 60% accuracy on the test dataset which is a promising result.

The wrong classifications show a common pattern regardless of their ground truth category: they cannot be classified by a human. Either because the object is not visible due to zero Doppler noise, or because only a small part of a large object is visible to the RADAR and therefore poorly reflects the signal (a large object will be wrongly classified as a small one).

Real raw RADAR data are necessary to take into account the entire complexity of the representation. They also should be annotated to train models in a supervised manner. To this end, the next section will describe methods to generate RD with their entire complexity.

4.2 RADAR data generation

This section presents methods to generate RDs from natural images (camera). A dataset of complex urban scenes with paired raw RADAR data and natural images has been recorded and used for our experiments. However, the RADAR data are not directly annotated due to the complexity of the representation. Because the objects in the raw RADAR representations are difficult for a human to distinguish, manual annotation is expensive and of poor quality. In this work, we tried to approximate a transformation to map the natural image domain (source) to the RD domain (target). In other words, the model will learn to generate the signature of objects in the RADAR representation by recognizing them in the camera image. This transformation is learnt with deep neural networks by reconstructing the data of the target domain using the source domain as input. The motivation of this work is to find a common latent space between the two domains to transfer information from the source to the target. In this way, annotations in the natural images can be transferred to the raw RADAR representations without human supervision.

4.2.1 Dataset

The dataset was recorded in Canada in an urban environment. The acquisition setup consists of a 77GHz FMCW RADAR and a camera mounted on a stationary car. The RADAR uses the MIMO system configuration with 2 Tx and 4 Rx producing a total of $N_{\text{Tx}} \cdot N_{\text{Rx}} = 8$ virtual antennas. The dataset contains $26,085$ frames divided into 8 sequences for a total duration of 44 minutes. It consists of complex urban scenes with paired raw RADAR and natural images recorded at 10 FPS. In this work, we used the recorded natural images and RD representations which have a resolution of 616×514 and 128×64 respectively. Bounding boxes annotations are provided for $\frac{1}{10}$ of the frames but only for the natural images. They are classified between two categories: pedestrian and vehicle. Figure 4.2 shows examples of scenes with pedestrians and cars at a crossroad.

The dataset is split between the training (90%), the validation (5%) and the test (5%) datasets. The training split is deliberately large to be able to train deeper neural network

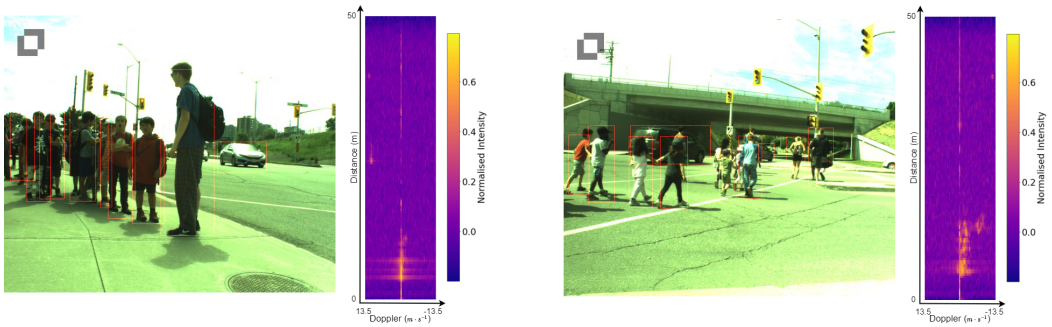


Figure 4.2: **Example of two scenes with pedestrians and moving cars**. For both scenes (a) and (b); Left: natural image of the scene with bounding boxes around the pedestrian and the moving cars; Right: Range-Doppler representation of the scene without annotation.

architectures. The next section will provide details about the range-Doppler representation.

4.2.2 Range-Doppler representation

The raw RADAR data are recorded in a 3D tensor in the frequency domain which is processed by an inverse FFT on each axis to obtain the RAD tensor in the temporal domain (see Section 2.2). Let $\mathbf{X}^{\text{RAD}}$ be the complex RAD tensor. The RD is obtained by aggregating the tensor on the second axis following Equation 2.19. This method aims to reduce the global noise on the representation as presented in Section 2.2.

The raw RADAR data is prone to a high level of noise, partly due to the speckle noise as detailed in Section 2.2.2. It is a noise inherent to the RADAR sensor covering the entire representation. While working on the reconstruction of the RD representation, it is helpful to estimate the distribution of the noise that the model will need to overcome. For a given sequence, the speckle noise is extracted by selecting an image patch in the same spatial position, over $1,050$ frames. The patch has been chosen so that almost no signal appears in the area.

Figure 4.3 shows the distributions of the background with different methods to aggregate the data. The data follows the model proposed by [Goodman 2007]: the original signal has a decreasing exponential distribution and the log-signal has a Fisher-Tippett distribution. The speckle aggregated in the third dimension of the tensor follows a Gamma distribution. Finally, after applying Equation 2.19 (averaging and log-transform), the speckled background follows a Fisher-Tippett distribution approximated by a Gaussian distribution estimated as $\mathcal{N}(35.38, \sqrt{2.82})$. Details about the speckle and its distributions are provided in Section 2.2.2. The following section will explain the methods and experiments conducted on the presented dataset.

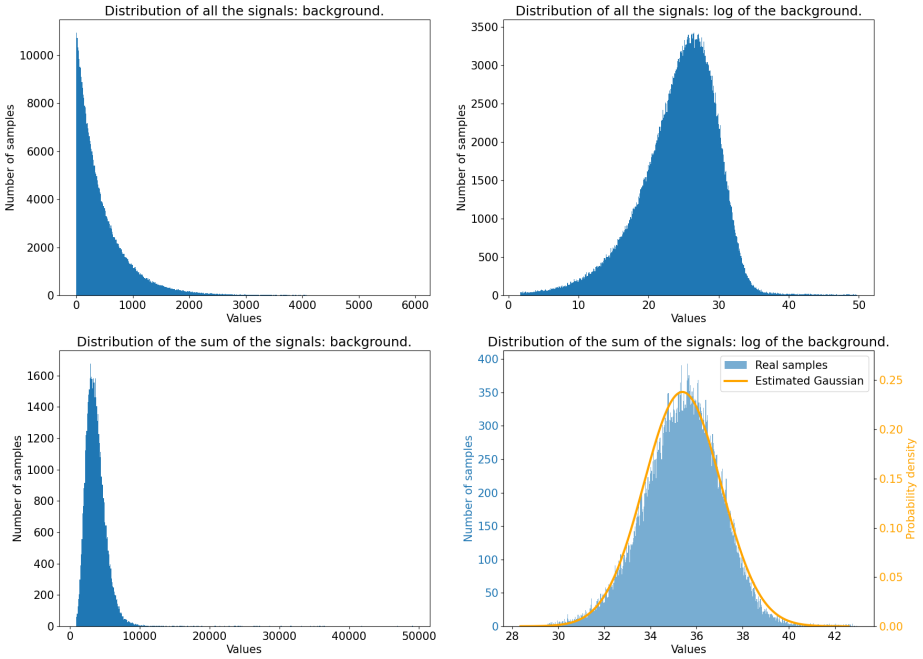


Figure 4.3: **Distributions of the background in RADAR data**. Distribution of the background for a given patch in a sequence before applying Equation 2.19. These sample distributions follow the model proposed by [Goodman 2007]; they are described in Section 2.2.2. Top left: distribution of the signal values (decreasing exponential distribution). Top right: distribution of the logarithm of the signal values (Fisher-Tippett distribution). Bottom left: distribution of the signal values summed over the antennas axis (Gamma distribution). Bottom right: distribution of the signal values after applying Equation 2.19 (blue) and distribution of random samples of an estimated Gaussian distribution (orange).

4.2.3 Methods and Experiments

This section describes the methods we explored to generate RD representations from natural images. We modified them to propose adapted architectures for RD reconstruction from sequence of natural images. The results of the experiments are presented in Table 4.3. Qualitative results of RD generation are compared in Figure 4.4.

The RD representation contains both the distance and the relative velocity of the reflectors in a scene w.r.t. the RADAR. At least two images are required to estimate the velocity of an object. The sequences are truncated to choose varying window sizes of natural images. Our experiments have shown that considering a short sequence of three consecutive images to generate an RD leads to better performances without incurring a significant computational cost (especially when considering 3D convolutions). In the following experiments, the images of the short sequence are stacked in the depth dimension to train neural network architectures.

Let $\mathcal{S}$ be the input domain (any sequence of three consecutive natural images), and $\mathcal{R}$ the output domain of RD representations. Our goal is to determine a function f_θ, parameterized by θ, which carries out this domain conversion, such as:

$$\begin{array}{rccc} f_\theta : & \mathcal{S} & \longrightarrow & \mathcal{R} \\ & \mathbf{x}_s & \longmapsto & \mathbf{x}_r. \end{array} \tag{4.3}$$

Note that the dimension of $\mathbf{x}_s$ is much larger than $\mathbf{x}_r$ ($\mathcal{S} \gg \mathcal{R}$). In these experiments, $\mathbf{x}_r$ is reconstructed using $\mathbf{x}_s$.

During training, the MSE is used as a loss function to reconstruct the RD representation: $\text{MSE} = \frac{1}{B_\text{R} \cdot B_\text{D}} \|\mathbf{x}_r - \mathbf{x}_r^*\|_2^2$ where $\mathbf{x}_r^* = f_\theta^*(\mathbf{x}_s)$, f_θ^* is the approximated reconstruction function and $B_\text{R} \cdot B_\text{D}$ the number of elements in the RD representation. The Mean Absolute Error (MAE) has also been used as a loss function but its primary goal is to compare the obtained results as an evaluation metric. It is written: $\text{MAE} = \frac{1}{B_\text{R} \cdot B_\text{D}} \|\mathbf{x}_r - \mathbf{x}_r^*\|_1$.

Well-known deep neural network methods and architectures have been considered. We adapted them in our experiments for RD reconstruction. In particular, we explored stacked convolutions with 2D or 3D kernels, AutoEncoder architectures, radar-based prior information to train the network, specific loss functions and generative approaches. The following paragraphs describes these methods in details.

Stacked 2D convolutions. This group of layers has been considered as a first approach while using down-samplings to reduce the size of the natural images and directly match the RD dimensions, noted $(B_\text{R} \times B_\text{D})$. A final 1D convolution layer is used to reduce the number of feature maps and generate the RD map. For this method, we adapted the VGG11 [Simonyan & Zisserman 2015] architecture in a VGG9 by replacing the last three fully-connected layers by a single 1D convolution layer to perform the reconstruction.

Stacked 3D convolutions. The interest of using 3D convolutions introduced by [Ji *et al.* 2012] is to take into account the temporal patterns in the sequence of natural images used as input. Deep architectures are not feasible due to the high computational

cost of 3D convolutions. We created a neural network architecture with 10 layers of 3D convolutions processing the sequence of images for RD reconstruction.

Convolutional AutoEncoder. This method [Masci *et al.* 2011] consists in processing the input data with 2D convolutions while deducing its dimensions with down-samplings.The latent space of feature maps is then up-sampled to recover the output dimensions and reconstruct the RD representation. A well-known extension of Convolutional AE is U-Net [Ronneberger *et al.* 2015]. Skip connections process the features learnt at different stages of the down-sampling pathway of the network and add the information to the paired stage of the up-sampling pathway. It helps to keep information even after reducing it into a low dimensional latent space. We modified the original U-Net architecture to match our input and output dimensions. The proposed AE architecture has the same down-sampling and up-sampling architectures as U-Net with 15 convolution layers in total while the U-Net have additional skip connections. In total, the U-Net contains 19 convolutions with skip connections. We also proposed a lighter version of the AE to reduce the training time ('Light-AE'). Reducing the number of filters leads to small improvements with three times fewer parameters.

Additional Noise Map. Our experiments have shown that the AE always tries to estimate the speckle noise distribution to reconstruct the RD representation. To leverage this observation, we proposed to stack the input data with a noise map randomly generated from the estimated background distribution detailed in Section 4.2.2. The noise map has the same dimensions as the input images. The noise map and the images are stacked in the channel dimension to provide prior information to the network on the speckle distribution. In the presented experiments, our proposed "Noise Map" approach leads to a significant gain of time during training to reach similar performances. It also improves results for a similar training time.

Custom Losses. Reflectors in a scene are represented by high intensity values in the RD representation. Previous methods have shown that a focus is made on the noise reconstruction while the shape of the object signature is difficult to recover. Considering the MSE as a loss leads to learn a global average of the intensity values and thus prioritizes the speckle noise. The reconstruction of the object's signature is important because the final objective of this work is to transfer object annotations from a domain to another. To focus the reconstruction on high intensity values, we have used a weighted Mean Absolute Error (wMAE). It considers the ground-truth intensity value as a weight which is applied pixel-wise in the MAE loss. The wMAE loss is defined as:

$$\text{wMAE} = \frac{1}{B_\text{R} \cdot B_\text{D}} \sum_{n=0}^{B_\text{R} \cdot B_\text{D} - 1} \mathbf{x}_{r,n} \left\| \mathbf{x}_{r,n} - \mathbf{x}^*_{r,n} \right\|_1 , \tag{4.4}$$

where $\mathbf{x}_{r,n}$ is the n-th element of the ground-truth RD representation in the $\mathcal{R}$ domain and $\mathbf{x}^*_r = f^*_\theta(\mathbf{x}_s)$ with f^*_θ the approximated reconstruction function.

To both take into account the high intensity value penalization and the global noise estimation, the wMAE_MSE loss has been experimented which combines both the wMAE and the MSE losses. The two terms are weighted by positive hyper-parameters λ_1 and λ_2. These values have not been explored yet, they are fixed to 1 in the experiments. The loss is defined as:

$$\text{wMAE\_MSE} = \frac{1}{B_\text{R} \cdot B_\text{D}} \left[\lambda_1 \sum_{n=0}^{B_\text{R} \cdot B_\text{D} - 1} \mathbf{x}_{r,n} \left\| \mathbf{x}_{r,n} - \mathbf{x}^*_{r,n} \right\|_1 + \lambda_2 \left\| \mathbf{x}_r - \mathbf{x}^*_r \right\|_2^2 \right]. \tag{4.5}$$

Generative Adversarial Network. This approach introduced by [Goodfellow *et al.* 2014] is usually used for data synthesis, in particular for natural images. It combines a set of two neural networks: a generator and a discriminator. For a given sample of data x, the first one generates a fake data sample using a random variable z as input (usually normally distributed). The second one is a binary classifier trying to detect if a sample is fake or real. Both networks are complementary, the generator tries to fool the discriminator with a high quality sample so it could not distinguish a real from a fake. Note that the generator takes only a random variable as input in the original approach, it can also take the initial sample x as input.

Both networks are trained with a minmax game using a cost function $V(G, D)$:

$$\min_G \max_D V(D, G) = \mathbb{E}_{x \sim p_\text{data}(x)}[\log D(x)] + \mathbb{E}_{z \sim p_z(z)}[\log(1 - D(G(z)))], \tag{4.6}$$

where G is the generator and D the discriminator. In their work, [Goodfellow *et al.* 2014] explain that minimizing $\log(1 - D(G(z)))$ leads to poor gradient at the beginning of the training since the generator is not efficient enough. In practice, $\log(D(G(z)))$ is maximized in the optimization process, it is called the Non-Saturating GAN Loss.

In the presented experiments, we proposed to learn a generator to create a fake RD using the natural image as input instead of a random noise. The discriminator takes either the generated or real RD and classifies it as fake or real. The loss function is composed of a MSE reconstruction loss for the generator and a BCE for the discriminator. The final objective is written:

$$\min_G \max_D \mathcal{L}_\text{GAN}(D, G) = \min_G \max_D \frac{1}{B_\text{R} \cdot B_\text{D}} \left\| \mathbf{x}_r - G(\mathbf{x}_s) \right\|_2^2 + V(D, G). \tag{4.7}$$

Note that the implemented GAN also uses patchGAN [Isola *et al.* 2017] for the discriminator classification. This method is described in the following paragraph.

We also considered a second approach called cGAN [Isola *et al.* 2017]. It consists in providing additional information to the networks as a condition. This condition helps the generator to better estimate to distribution of the generated samples. For a given sample (x, y), the generator tries to reconstruct y using the original data x and a random noise z. In this case, the condition x provides the distribution of the data in the source domain to help the generator in estimating the distribution of the data in the target domain. The dis-

Method	#Parameters	Loss	MSE Scores		MAE Scores	
			Validation	Test	Validation	Test
VGG-9	9.23M	MSE	8.08	7.74	1.69	1.67
3D Conv.	0.23M	MSE	9.10	8.74	1.83	1.81
AE	12.9M	MSE	4.76	4.84	1.50	1.50
U-Net [Ronneberger *et al.* 2015]	14.4M	MSE	5.22	5.16	1.52	1.52
AE + NoiseMap	12.9M	MSE	**4.52**	**4.51**	**1.48**	**1.48**
Light-AE	3.24M	MSE	4.64	4.63	1.49	1.49
Light-AE	3.24M	wMAE	5.16	5.16	1.50	1.49
Light-AE	3.24M	wMAE_MSE	5.16	5.17	1.50	1.50
Light-GAN	3.31M	$\mathcal{L}_{\text{GAN}}$	4.92	4.87	1.50	1.50
Light-cGAN	4.98M	$\mathcal{L}_{\text{cGAN}}$	5.14	5.12	1.49	1.48

Table 4.3: **Quantitative performances of Range-Doppler reconstruction**. Performances of the methods detailed in Section 4.2.3, trained and tested on the splits of the dataset presented in Section 4.2.1, regarding the MSE and MAE evaluation metrics.

criminator classifies either fake or real the generated or the real sample y while observing x. Considering Equation 4.8, the objective function is defined as:

$$\min_G \max_D V_x(D,G) = \mathbb{E}_{x,y}[\log D(x,y)] + \mathbb{E}_{x,z}[\log(1 - D(x, G(x,z)))]. \tag{4.8}$$

In their work, [Isola *et al.* 2017] described the patchGAN method to improve the discriminator consistency. Instead of directly classifying a sample as fake or real, each feature map of the last layer of the discriminator is classified in a category. The final discrimination result is obtained by aggregating the sub classifications on the feature maps. This way, the discriminator takes into account local pattern similarities. The MAE loss is used for the generator and the BCE loss (negative log likelihood) for the discriminator. The objective function is defined as:

$$\min_G \max_D \mathcal{L}_{\text{cGAN}}(D,G) = \frac{1}{B_{\text{R}} \cdot B_{\text{D}}} \left\| \mathbf{x}_r - G(\mathbf{x}_s) \right\|_1 + V_x(D,G). \tag{4.9}$$

In our experiments, we note 'Light-GAN' and 'Light-cGAN' the corresponding generative methods using a Light-AE architecture for the generator, and the original architectures proposed by [Goodfellow *et al.* 2014] and [Isola *et al.* 2017] for the discriminator.

Results. Quantitative results presented in Table 4.3 show that our proposed Convolutional AE an Light-AE architectures, inspired from U-Net and detailed in the previous paragraph, reached the best performances regarding the MSE and MAE metrics. In particular, our proposed AE processing an additional noise map generated from the speckle noise distribution obtained the higher reconstruction performances. Qualitative results on the test dataset presented in Section 4.2.1 are illustrated in Figure 4.4. The generated RD representations using AE architectures succeed to recover several object signatures while reconstructing the zero-Doppler reflections. However, the speckle noise is estimated as a global average leading to a smooth background of intensities.

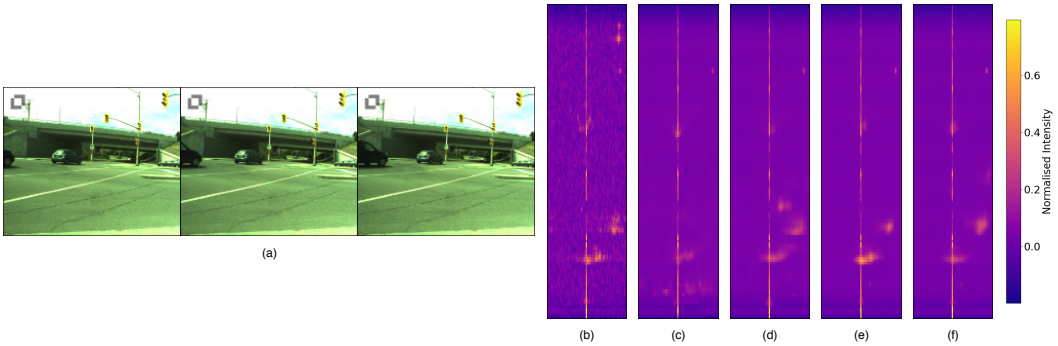


Figure 4.4: **Qualitative results of Range-Doppler generations**. (a) Consecutive natural images images of a scene with (b) the Range-Doppler ground-truth associated to the image in the center. Range-Doppler generated from the sequence of natural images using: (c) VGG9, (d) AutoEncoder, (e) Light AutoEncoder, (f) AutoEncoder with noise map (ours).

4.2.4 Discussions

This section has described several methods to generate Range-Doppler representations using natural images. The presented experiments have shown that our Noise Map method integrating a map of generated speckle noise leads to increase the performances while reducing the training time. GAN and cGAN architectures have produced interesting results but further experiments should be conducted to reach better performances. The selected methods did not generate high quality RD so the annotation transfer from the image domain to the RADAR domain is not usable with this approach. In the next section, a dataset is introduced with non-urban scenes and a different pipeline to annotate the RADAR representations in a semi-supervised manner.

4.3 CARRADA dataset

High quality perception is essential for autonomous driving systems. To reach the accuracy and robustness that are required by such systems, several types of sensors must be combined. Currently, mostly cameras and lidar are deployed to build a representation of the world around the vehicle. While RADAR sensors have been used for a long time in the automotive industry, they are still under-used for autonomous driving despite their appealing characteristics (notably, their ability to measure the relative speed of obstacles and to operate even in adverse weather conditions). To a large extent, this situation is due to the relative lack of automotive datasets with real RADAR signals that are both raw and annotated as detailed in Section 3.2. In this section, CARRADA is introduced, a dataset of synchronized camera and RADAR recordings with Range-Angle-Doppler annotations. A semi-automatic annotation approach is also presented, which was used to annotate the dataset, and a RADAR semantic segmentation baseline evaluated on several metrics. A sample of camera and RADAR data with generated annotation is illustrated in Figure 4.5.

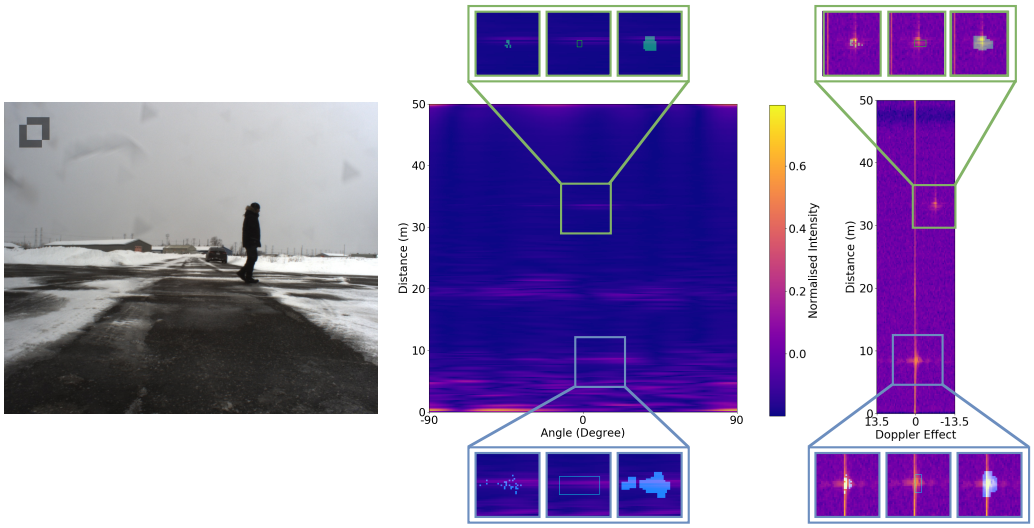


Figure 4.5: **A scene from CARRADA dataset, with a pedestrian and a car**. (a) Video frame provided by the frontal camera, showing a pedestrian at approximately 8m from the sensors and a car in the background at approximately 33m; (b-c) RADAR signal at the same instant in Range-Angle and Range-Doppler representation respectively. Three types of annotations are provided: sparse points, bounding boxes and dense masks. The blue squares correspond to the pedestrian and the green ones to the car. Note that the Range-Angle view illustrated in (b) has been obtained using a maximum aggregation and a log-transform (Eq. 2.18) while the Range-Doppler view in (c) is computed using Equation 2.19.

Both our code and dataset are available online [1]. This section presents a work mainly inspired from our article published at the International Conference on Pattern Recognition (ICPR) [Ouaknine *et al.* 2020].

4.3.1 Dataset

The dataset has been recorded in Canada on a test track to reduce environmental noise. The acquisition setup consists of an AWR1843-BOOST[2] FMCW RADAR and a camera mounted on a stationary car. The RADAR uses the MIMO system configuration with 2 Tx and 4 Rx producing a total of 8 virtual antennas. The parameters and specifications of the sensor are provided in Table 4.4. The image data recorded by the camera and the RADAR data are synchronized to have the same frame rate in the dataset. The sensors are also calibrated to have the same Cartesian coordinate system. The image resolution is 1238×1028 pixels. The Range-Doppler and Range-Angle representations are respectively stored in 2D matrices of size 256×64 ($B_R \times B_D$) and 256×256 ($B_R \times B_A$).

One or two objects are moving in the scene at the same time with various trajectories to

[1] https://github.com/valeoai/carrada_dataset
[2] https://www.ti.com/tool/AWR1843BOOST

Parameter	Value
Frequency	77 Ghz
Sweep Bandwidth	4 Ghz
Maximum Range	50 m
FFT Range Resolution	0.20 m
Maximum Radial Velocity	13.43 m/s
FFT Radial Velocity Resolution	0.42 m/s
Field of View	180°
FFT Angle Resolution	0.70°
Number of Chirps per Frame	64
Number of Samples per Chirp	256

Table 4.4: **Parameters and settings of the RADAR sensor**.

Parameter	Value
Number of sequences	30
Total number of instances	78
Total number of frames	12666 (21.1 min)
Maximum number of frames per seq.	1017 (1.7 min)
Minimum number of frames per seq.	157 (0.3 min)
Mean number of frames per seq.	422 (0.7 min)
Total number of annotated frames with instance(s)	7193 (12.0 min)

Table 4.5: **Statistics of the CARRADA dataset**.

simulate urban driving scenarios. The distribution of these scenarios across the dataset is shown in Figure 4.6. The objects are moving in front of the sensors: approaching, moving away, going from right to left or from left to right (see examples in Figure 4.12). Each object is an *instance* tracked in the sequence. The distribution of mean radial velocities for each object category is provided in Figure 4.7, while other global statistics about the recordings can be found in Table 4.5.

Object signatures are annotated in both RA and RD representations for each sequence. Each instance has an identification number, a category and a localization in the data. Three types of annotations for localization are provided: sparse points, boxes and dense masks. The next section will describe the pipeline used to generate them.

4.3.2 Pipeline for annotation generation

Automotive RADAR representations are difficult to understand compared to natural images. Objects are represented by shapes with varying sizes carrying physical measures. It is not a trivial task to produce good quality annotations on this data. This section details a semi-automatic pipeline based on video frames to provide annotations on RADAR representations.

4.3.2.1 From vision to physical measurements

The camera and RADAR recordings are synchronized. Visual information in the natural images is used to obtain physical prior knowledge about an instance as well as its category. The real-world coordinates of the instance and its radial velocity are estimated generating the annotation in the RADAR representation. This first step instantiates a tracking pipeline propagating the annotation in the entire RADAR sequence.

Each video sequence is processed by a Mask R-CNN [He *et al.* 2017] model providing both semantic segmentation and bounding box predictions for each detected instance. Both are required for our pipeline to compute the center of mass of the object and to track it. Instance tracking is performed with the Simple and Online Real time Tracking (SORT) algorithm [Bewley *et al.* 2016]. This light-weight tracker computes the overlap between the predicted boxes and the tracked boxes of each instance at the previous frame. The

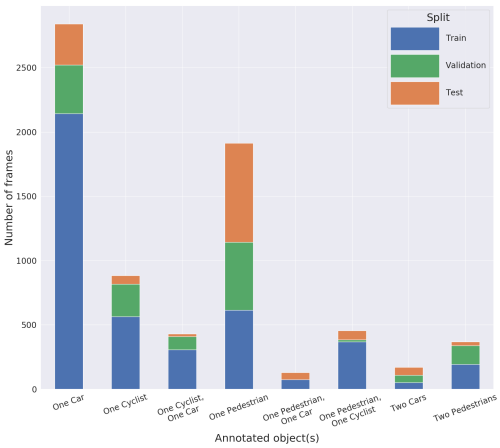


Figure 4.6: **Object distribution across CARRADA**. Distribution of the eight object configurations present in the dataset, expressed as frame numbers across the three parts of the proposed split.

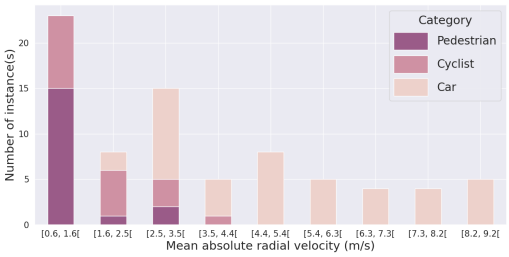


Figure 4.7: **Distribution of radial velocities for all categories**. For each annotated instance, its absolute radial velocity in m/s is averaged, over its sparse annotations in each frame and over time, and one histogram is built for each object class. Note that annotated velocities are actually signed (negative when the object if moving away and positive when it approaches the radar).

selected boxes are the most likely to contain the same instance, *i.e.* the boxes with the highest overlap.

The center of mass of each segmented instance is projected on the bottom-most pixel coordinates of the segmentation mask. This projected pixel localized on the ground is considered as the reference point of the instance. Using the intrinsic and extrinsic parameters of the camera, pixel coordinates of a point in the real-world space are expressed as:

$$s\ \mathbf{p} = A\ B\ \mathbf{c}, \tag{4.10}$$

where $\mathbf{p} = [p_x, p_y, 1]^\top$ and $\mathbf{c} = [c_x, c_y, c_z, 1]^\top$ are respectively the pixel coordinates in the image and the real-world point coordinates, s a scaling factor, and A and B are the intrinsic and extrinsic parameters of the camera defined as:

$$A = \begin{bmatrix} f_x & 0 & a_x \\ 0 & f_y & a_y \\ 0 & 0 & 1 \end{bmatrix}, B = \begin{bmatrix} r_{11} & r_{12} & r_{13} & m_1 \\ r_{21} & r_{22} & r_{23} & m_2 \\ r_{31} & r_{32} & r_{33} & m_3 \end{bmatrix}. \tag{4.11}$$

Using Equation 4.11, one can determine $\mathbf{c}$ knowing $\mathbf{p}$ with a fixed value of elevation.

Given the time interval δt separating two frames $t - \delta t$ and t, the velocity vector $\mathbf{v}^t$ is defined as:

$$\mathbf{v}^t = \mathbf{c}^t - \mathbf{c}^{t-\delta t}, \tag{4.12}$$

where $\mathbf{c}^t$ is the real-world coordinate in frame t. The time interval chosen in practice is $\delta t = 1$ second.

The Doppler effect recorded by the RADAR is the radial velocity of the instance re-

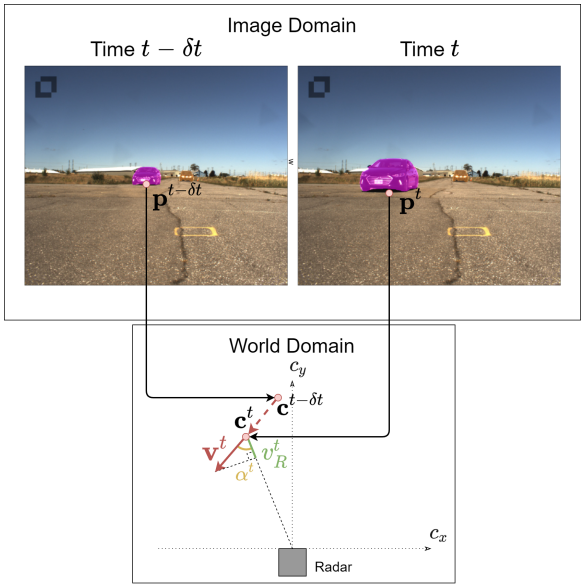


Figure 4.8: **Estimation of the radial velocity from natural images**. The space (c_x, c_y, c_z) defines real-world coordinates considering the RADAR as origin, c_z is fixed to zero. Points in the real-world domain $\mathbf{c}^{t-\delta t}$ and $\mathbf{c}^{t}$ (bottom) are estimated using the points in the pixel domain $\mathbf{p}^{t-\delta t}$ and $\mathbf{p}^{t}$ (top). The velocity vector $\mathbf{v}^t$ is estimated with the real-world points. The radial velocity v_R^t of the object at time t corresponds to the projection of its velocity vector on the straight line between the RADAR of the object.

flecting the signal. The radial velocity v_R^t at a given frame t is defined as:

$$v_R^t = \cos \alpha^t \, \|\mathbf{v}^t\|, \tag{4.13}$$

where α^t is the angle formed by $\mathbf{v}^t$ and the straight line between the RADAR and the instance. The quantization of the radial velocity is illustrated in Figure 4.8. This way, each instance detected in the frame is characterized by a feature point $I^t = [\mathbf{c}^t, v_R^t]^\top$. This point will be projected in a RADAR representation to annotate the raw data and track it in this representation.

4.3.2.2 DoA clustering and centroid tracking

The RA representation is a RADAR scene in polar coordinates. Its transformation into Cartesian coordinates is called DoA. Points are filtered by a CFAR algorithm [Rohling 1983] keeping the highest intensity values while taking into account the local relation between points. The DoA is then a sparse point cloud in a 2D coordinate space similar to a BEV representation. The representation is enriched using the recorded Doppler for each point as the third axis of the point cloud. The 3D point cloud combines the Cartesian coordinates of the reflected point and its Doppler value. This helps to distinguish the signature boundaries of different objects. The feature point I^t is projected in this space and assigned to a cluster of points considered as the reflection of the targeted instance. It

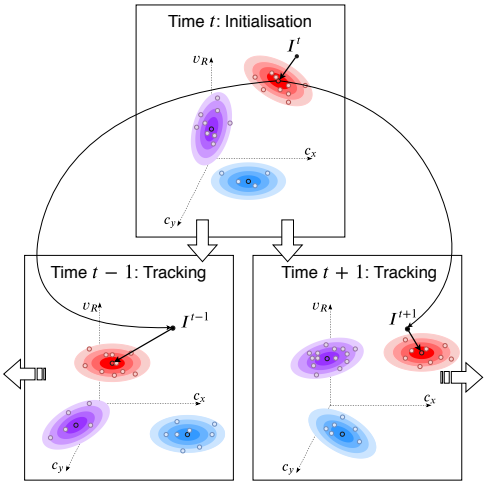


Figure 4.9: **Tracking of the Mean Shift cluster to propagate the annotation in the sequence**. The Mean Shift algorithm used with the bandwidth selection method is applied to the DoA-Doppler representation at time t. The estimated point I^t, using the computer vision pipeline and corresponding to the tracked object, is associated to its closest cluster. The centroid of this cluster considered as I^{t+1} and I^{t-1} in the next and previous DoA-Doppler frames is tracked iteratively.

is then tracked in the past and future using the following process, illustrated in Figure 4.9. This initialisation requires a human supervision to validate the matching between I^t and the cluster of RADAR points corresponding to the object before the tracking. This supervision could be avoided by considering an automatic validation between the camera and the RADAR frames, it will be explored in a future work.

At a given timestamp chosen by the user, a 3D DoA-Doppler point cloud is clustered using the Mean Shift algorithm [Comaniciu & Meer 2002]. Let $\{\mathbf{x}_0, \cdots, \mathbf{x}_{n-1}\}$ be a point cloud of n points. For a given starting point, the algorithm iteratively computes a weighted mean of the current local neighborhood and updates the point until convergence. Each iteration reads:

$$\mathbf{x} \leftarrow \frac{\sum_{i=0}^{n-1} \mathbf{x}_i K(\mathbf{x}; \mathbf{x}_i, \sigma)}{\sum_{i=0}^{n-1} K(\mathbf{x}; \mathbf{x}_i, \sigma)}, \tag{4.14}$$

where $K(\mathbf{x}; \mathbf{x}_i, \sigma) = \mathcal{N}(\left\| \frac{\mathbf{x}-\mathbf{x}_i}{\sigma} \right\|; 0, 1)$ is the multivariate spherical Gaussian kernel with bandwidth σ, centered at $\mathbf{x}_i$. All initial points leading to close final locations at convergence are considered as belonging to the same cluster.

Mean-Shift clustering is sensitive to the bandwidth parameter σ. Its value should depend on the point cloud distribution and it is usually defined with prior knowledge about the data. In this application, it is not straightforward to group points belonging to the same object in the DoA-Doppler point cloud representation. The number of points and their distribution depend on the distance and the surface of reflectivity of the target. Moreover, these characteristics change during a sequence while the instance is moving in front of the RADAR. In the work of [Bugeau & Pérez 2007], the authors proposed to compute several

times the Mean-Shift algorithm with ordered bandwidth values. They finally selected the optimal bandwidth by comparing the distribution of the clusters. Our method applied the same procedure to point clouds with multiple instances. It automatically finds an optimal bandwidth for each instance contained in each point cloud. We proposed additional metrics to compare the cluster distributions, they are described in the following paragraphs.

For a given DoA-Doppler point cloud, the closest cluster to the feature point I^t is associated to an instance. Let $\sigma_b \in \{\sigma_0, \cdots, \sigma_{B-1}\}$ be a bandwidth in a range of B ordered values. A Mean-Shift algorithm noted MeanShift(σ_b) selects the closest cluster $\mathcal{C}_b$ to I^t containing n_b points. After running the algorithm with all bandwidth values, $\{\mathcal{C}_0, \cdots, \mathcal{C}_{B-1}\}$ clusters are found with their corresponding optimal bandwidths.

The following paragraphs will detail methods to select the optimal bandwidth σ_{b^*} regarding the distribution and the stability of the clustered point clouds. Once the optimal bandwidth is found, the closest cluster to I^t using MeanShift(σ_{b^*}) is considered as belonging to the targeted instance. The points I^{t+1} and I^{t-1} are assigned with the centroid of this cluster. The process is then iterated in the previous and next frames to track the center of the initial cluster until the end of the sequence.

Log-Ratio The further away the object, the smaller its signature in the RADAR representation and the smaller its DoA point cloud. The proposed Log-Ratio method defines the bandwidth for the Mean Shift algorithm as a function of the distance of the centroid of the DoA cluster tracked from the previous or next RADAR frame [3]. Let d^{max} and d_{t-1} be respectively the maximum distance of the RADAR detection (50 meters in our case) and the distance to the centroid of the tracked DoA-Doppler point cloud at $t-1$ (or $t+1$ depending on tracking direction). At time t, the optimal bandwidth $\sigma_{b^*}(d_{t-1})$ is defined as:

$$\sigma_{b^*}(d_{t-1}) = \frac{1+\log(1+d^{\text{max}})}{1+\log(1+d_{t-1})} = \begin{cases} 1 & \text{if} \quad d_{t-1} = d^{\text{max}} \\ 1+\log(1+d^{\text{max}}) & \text{if} \quad d_{t-1} = 0 \\ \frac{1+\log(1+d^{\text{max}})}{1+\log(1+d_{t-1})} & \text{otherwise.} \end{cases} \tag{4.15}$$

Determinant In this method, the optimal bandwidth σ_{b^*} is selected by comparing the stability of the determinant between the selected clusters. For each $b \in \{0, \cdots, B-1\}$ ordered values and a cluster $\mathcal{C}_b = \{\mathbf{x}_0, \cdots, \mathbf{x}_{n_b-1}\}$ of n_b points, its Gaussian distribution $\mathcal{N}(\widehat{\mu}_b, \widehat{\Sigma}_b)$ is estimated with expectation $\widehat{\mu}_b = \frac{1}{n_b}\sum_{i=0}^{n_b-1}\mathbf{x}_i$ and variance $\widehat{\Sigma}_b = \frac{1}{n_b-1}\sum_{i=0}^{n_b-1}(\mathbf{x}_{i\cdot} - \widehat{\mu}_b)(\mathbf{x}_{i\cdot} - \widehat{\mu}_b)^\top$. For each bandwidth b, the determinant $|\widehat{\Sigma}_b| = \text{Det}(\widehat{\Sigma}_b)$ of the variance-covariance matrix of the corresponding Gaussian distribution is computed. Using these determinants from the fitted distributions as signatures, the bandwidth σ_{b^*} is selected by choosing the one which is the most "stable" with respect to a varying bandwidth:

$$b^* = \underset{b\in\{1,\cdots,B-2\}}{\text{argmin}} \; (|\widehat{\Sigma}_b| - |\widehat{\Sigma}_{b-1}|) + (|\widehat{\Sigma}_{b+1}| - |\widehat{\Sigma}_b|). \tag{4.16}$$

[3] It depends on the direction of the annotation propagation in the RADAR sequence, either it is in the future or in the past depending on the initialization frame.

Number of points The optimal bandwidth σ_{b^*} is selected by comparing the stability of the number of points between the selected clusters. For each $b \in \{0, \cdots, B-1\}$ ordered values, the cluster $\mathcal{C}_b \in \{\mathcal{C}_0, \cdots, \mathcal{C}_{B-1}\}$ corresponds to the closest cluster regarding the tracked centroid in a sequence using MeanShift(σ_b). The bandwidth σ_{b^*} is selected by choosing the one which is the most "stable" with respect to a varying bandwidth:

$$b^* = \underset{b \in \{1, \cdots, B-2\}}{\operatorname{argmin}} (\#\mathcal{C}_b - \#\mathcal{C}_{b-1}) + (\#\mathcal{C}_{b+1} - \#\mathcal{C}_b), \tag{4.17}$$

where $\#\mathcal{C}_b = \sum_{x \in \mathcal{C}_b} 1$ is the number of points in the cluster $\mathcal{C}_b$.

Jensen-Shannon divergence The optimal bandwidth σ_{b^*} is selected by comparing the stability of the probability distribution of the points between the selected clusters. For each $b \in \{0, \cdots, B-1\}$ ordered values, the probability distribution p_b estimated with the n_b points of the cluster $\mathcal{C}_b = \{\mathbf{x}_0, \cdots, \mathbf{x}_{n_b-1}\}$ is the Gaussian distribution $\mathcal{N}(\widehat{\mu}_b, \widehat{\Sigma}_b)$ with expectation $\widehat{\mu}_b = \frac{1}{n_b} \sum_{i=0}^{n_b-1} \mathbf{x}_i$ and variance $\widehat{\Sigma}_b = \frac{1}{n_b-1} \sum_{i=0}^{n_b-1} (\mathbf{x}_{i\cdot} - \widehat{\mu}_b)(\mathbf{x}_{i\cdot} - \widehat{\mu}_b)^\top$. Using these fitted distributions, the bandwidth σ_{b^*} is selected by choosing the one which is the most "stable" with respect to a varying bandwidth:

$$b^* = \underset{b \in \{1, \cdots, B-2\}}{\operatorname{argmin}} \Big[JS\big(p_b \| p_{b-1}\big) + JS\big(p_b \| p_{b+1}\big) \Big], \tag{4.18}$$

where JS is the Jensen-Shannon divergence [Endres & Schindelin 2003]. This is a proper metric derived from Kullback-Leibler (KL) divergence [Kullback & Leibler 1951] as

$$JS(p\|q)^2 = \frac{KL(p\|\frac{p+q}{2}) + KL(q\|\frac{p+q}{2})}{2}, \tag{4.19}$$

where $KL(p\|\frac{p+q}{2}) = \sum_i p(i) \frac{2p(i)}{p(i)+q(i)}$, for two discrete probability distributions p and q. The Jensen-Shannon (JS) divergence measure the similarity between q and p since it is symmetric and has a finite value, contrary to the Kullback-Leibler divergence.

The presented methods have been tested and qualitatively compared on the CARRADA dataset. The JS divergence used to quantify the stability between the probability distributions of compared clustered has reached the best results. It succeeds in including a large number of RADAR points belonging to the object through time. An example comparing the proposed methods is illustrated in Figure 4.10. The temporal consistency of the bandwidth selection using the JS divergence is also highlighted in Figure 4.12.

4.3.2.3 Projections and annotations

We recall that $\mathcal{C}_{b^*}$ is the cluster associated to the point I^t at time t using MeanShift(σ_{b^*}), where σ_{b^*} is the estimated optimal bandwidth. This cluster is considered as belonging to the tracked object. A category is associated to it by using the segmentation model on the image (Section 4.3.2.1). The points are projected onto the RD representation using the radial velocity and the distance is computed with the real-world coordinates. They are also projected onto the RA representation by converting the Cartesian coordinates to polar

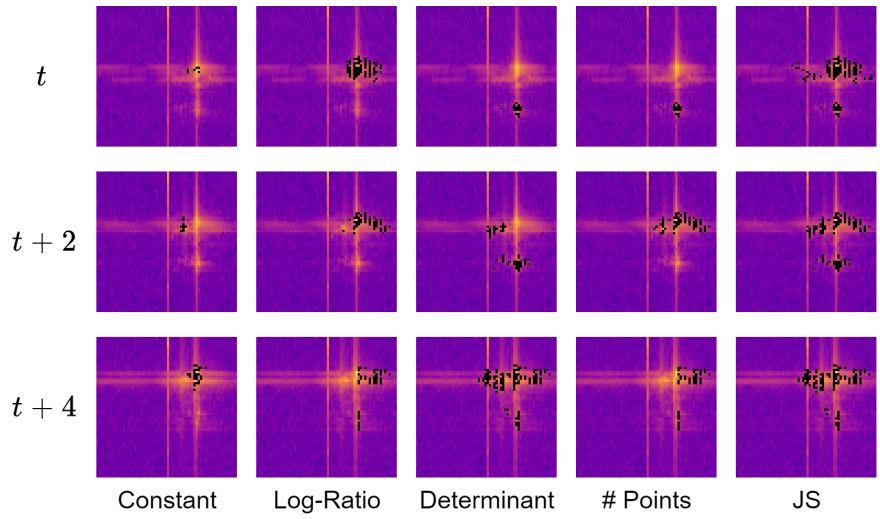


Figure 4.10: **Comparison of bandwidth selection methods**. Qualitative results of the bandwidth selection methods applied to the Mean Shift algorithm presented in Section 4.3.2.1. The algorithm is applied at each timestamp of the RADAR sequence in order to track the cluster corresponding to the object. Results are illustrated for a sequence between timestamps t and $t+4$. The black points correspond to the tracked cluster of points in the DoA-Doppler space. They are projected in the Range-Doppler representation cropped for visualisation purpose.

coordinates.

Let f_D be the function which projects a point from the DoA-Doppler representation into the RD representation. Similarly, we denote with f_A the projection into the RA representation. The sets of points $\mathcal{M}_D = f_D(\mathcal{C}_b)$ and $\mathcal{M}_A = f_A(\mathcal{C}_b)$ correspond, respectively, to the RD and RA representations of $\mathcal{C}_b$. They are called the sparse-point annotations.

The bounding box of a set of points in $\mathbb{R}^2$ (either from $\mathcal{M}_D$ or $\mathcal{M}_A$) is defined as a rectangle parameterized by $\{(x_{\min}, y_{\min}), (x_{\max}, y_{\max})\}$ where $x_{\min}$ is the minimum x-coordinate of the set, $x_{\max}$ is the maximum, and similarly for the y-coordinates.

Mathematical morphology has been experimented to expand the sparse annotation creating a dense mask. Let $\mathcal{E} \subset \mathbb{R}^2$ be a structuring element, the **dilation** of $\mathcal{M}$ by $\mathcal{E}$ is defined as $\mathcal{M} \oplus \mathcal{E} = \cup_{e \in \mathcal{E}} \mathcal{M}_e$, where $\mathcal{M}_e$ is the translation of $\mathcal{M}$ by e. The **erosion** of $\mathcal{M}$ by $\mathcal{E}$, reducing the shape of $\mathcal{M}$ as opposed to the dilation, is written $\mathcal{M} \ominus \mathcal{E} = \cap_{e \in \mathcal{E}} \mathcal{M}_{-e}$, where $\mathcal{M}_{-e}$ is the translation of $\mathcal{M}$ by $-e$. Finally, the **closing** of $\mathcal{M}$ by $\mathcal{E}$ is the erosion of the dilation of this set. It is noted $\mathcal{M} \bullet \mathcal{E} = (\mathcal{M} \oplus \mathcal{E}) \ominus \mathcal{E}$.

The dense mask annotation is obtained by dilating the sparse annotated set with a circular structuring element: given the sparse set of points $\mathcal{M} = \{(x_0, y_0), \ldots, (x_{N-1}, y_{N-1})\}$, the associated dense mask is the set of discrete coordinates in $\cup_{i=0}^{N-1} \mathcal{B}_r(x_i, y_i)$, where $\mathcal{B}_r(x, y)$ is the disk of radius r centered at (x, y). It is called the "**Dilation/Circle**" method. We compared it with three other combinations of operations: a dilation with a cross structuring element (Dilation/Cross), the Dilation/Circle followed by a closing with a square structuring element (Dilation/Circle-Closing/Square) and the Dilation/Circle fol-

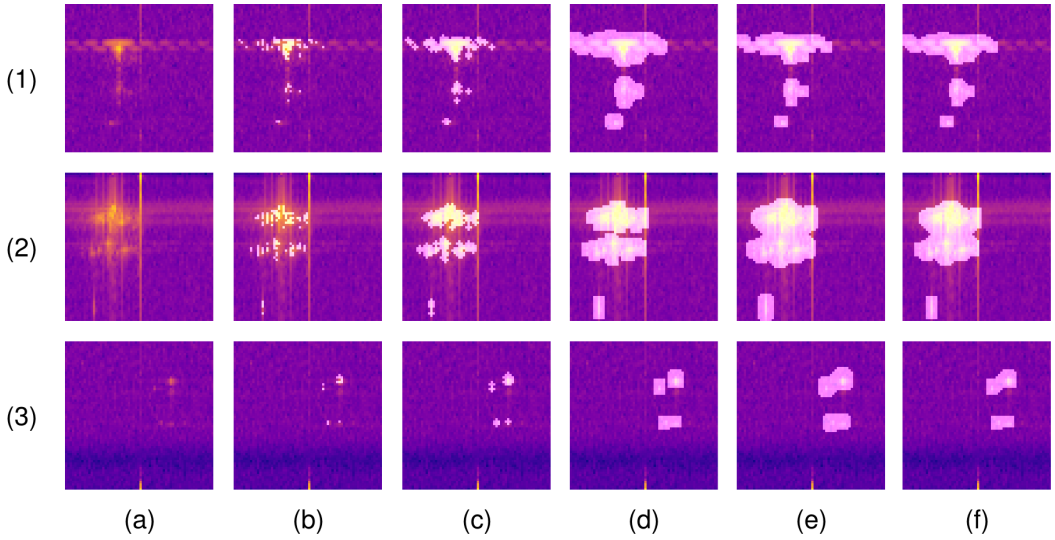


Figure 4.11: **Comparison of methods for dense mask annotation generation**. For each (1-3) independent example, the (a) source data is an object signature in a cropped Range-Doppler. Using the tracking pipeline and the projections presented in Sections 4.3.2.2 and 4.3.2.3, the cluster (b) is projected back in the representation, generating the sparse annotation. Qualitative results of the following methods are illustrated: (c) Dilation/Cross, (d) Dilation/Circle, (e) Dilation/Circle-Closing/Square, (f) Dilation/Circle-Closing/Square-Erosion/Square.

lowed by a closing and an erosion, both with a square structuring element (Dilation/Circle-Closing/Square-Erosion/Square). Note that the structuring elements are centered in a 3×3 binary patch. The cross structuring element has five 1 values and the square has nine 1 values.

These combinations of mathematical morphology operations are compared to create the dense mask from the sparse annotation. Qualitative results are illustrated in Figure 4.11. The Dilation/Circle method, used on its own, reached convincing results by covering homogeneously the shape of the object signature while including fewer noise bins than other methods for small objects.

In the following section, a baseline is proposed for RADAR semantic segmentation trained and evaluated on the annotations detailed above.

4.3.3 Semantic segmentation baseline

We proposed a baseline for semantic segmentation using RD or RA RADAR representation to detect and classify annotated objects. FCNs [Long *et al.* 2015] are used here to learn features at different scales by processing the input data with convolutions and down-sampling. Feature maps from convolutional layers are up-sampled with up-convolutions to recover the original input size. Each bin of the output segmentation mask is then classified. The particularity of FCN is that is uses skip connections from features learnt at different levels of the network to generate the final output. We denote with FCN-32s a network

Data	Model	IoU						PP						PR					
		Background	Pedestrian	Cyclist	Car	mIoU	hIoU	Background	Pedestrian	Cyclist	Car	mPP	hPP	Background	Pedestrian	Cyclist	Car	mPR	hPR
RD	FCN-32s	99.6 (—)	16.8 (—)	3.2 (—)	27.5 (—)	36.8 (—)	8.1 (—)	99.6 (—)	69.4 **(17.3)**	5.4 (1.0)	64.2 **(17.2)**	59.7 (11.9)	13.8 (2.2)	99.9 (—)	20.3 (28.3)	6.7 (8.1)	32.3 (47.5)	39.8 (28.0)	13.3 (14.0)
	FCN-16s	99.6 (—)	28.9 (—)	7.2 (—)	42.1 (—)	44.5 (—)	17.2 (—)	99.7 (—)	64.7 (15.1)	18.0 (4.7)	67.6 (17.0)	62.5 (12.3)	40.3 (8.6)	99.9 (—)	39.2 (50.9)	10.7 (16.9)	54.1 (74.1)	51.0 (47.3)	23.4 (27.5)
	FCN-8s	99.7 (—)	**45.2** (—)	**15.5** (—)	**51.3** (—)	**52.9** (—)	**34.2** (—)	99.8 (—)	**72.3** (15.5)	**35.2** **(9.6)**	**69.8** (17.0)	**69.3** **(14.0)**	**59.8** **(13.1)**	99.9 (—)	**55.0** **(76.4)**	**22.1** **(35.7)**	**66.8** **(88.9)**	**60.9** **(67.0)**	**44.3** **(56.7)**
RA	FCN-32s	99.8 (—)	0.0 (—)	0.0 (—)	14.2 (—)	28.5 (—)	0.0 (—)	99.9 (—)	18.8 (6.1)	0.0 (0.0)	**69.3** **(13.7)**	47.0 (6.6)	0.0 (0.0)	100.0 (—)	0.0 (0.1)	0.0 (0.0)	15.5 (24.6)	28.9 (8.2)	0.0 (0.0)
	FCN-16s	99.8 (—)	0.9 (—)	0.0 (—)	13.7 (—)	28.6 (—)	0.0 (—)	99.9 (—)	39.8 (5.2)	**0.2** (0.0)	68.0 (12.2)	**52.0** (5.8)	**0.9** (0.0)	100.0 (—)	0.9 (1.7)	0.0 (0.0)	15.4 (22.7)	29.1 (8.1)	0.0 (0.0)
	FCN-8s	99.9 (—)	**5.5** (—)	0.0 (—)	**25.1** (—)	**32.6** (—)	**0.1** (—)	99.9 (—)	**42.2** **(10.0)**	0.1 **(0.1)**	65.4 (12.4)	51.9 **(7.5)**	0.6 **(0.2)**	100.0 (—)	**6.3** **(11.3)**	0.0 **(0.1)**	**30.0** **(45.5)**	**34.1** **(19.0)**	**0.1** **(0.3)**

Table 4.6: **Semantic segmentation performances (%) on the test dataset for Range-Doppler (RD) and Range-Angle (RA) representations**. Models are trained on dense mask annotations and evaluated on both dense mask (top values) and sparse points (bottom values in parentheses) annotations. Results are evaluated with IoU, PP and PR. Metrics are computed by category and aggregated with both arithmetical (m) and harmonic (h) means. Lines (—) replacing values indicate non-applicable metrics, for example IoU results on sparse annotations.

where the output mask is generated only by up-sampling and processing feature maps with $1/32$ resolution of the input. Similarly, FCN-16s is a network where $1/32$ and $1/16$ resolution features maps are used to generate the output mask. In the same manner, FCN-8s fuses $1/32$, $1/16$ and $1/8$ resolution feature maps for output prediction.

The models are trained to recover dense mask annotations with four categories: *background*, *pedestrian*, *cyclist* and *car*. The background corresponds to speckle noise, sensor noise and artefacts which are covering most of the raw RADAR data. Parameters are optimized for 100 epochs using a categorical cross entropy loss function and the Adam optimizer [Kingma 2015] with the recommended parameters ($\beta_1 = 0.9$, $\beta_2 = 0.999$ and $\varepsilon = 1 \cdot 10^{-8}$). The batch size is fixed to 20 for the RD representation and to 10 for the RA representation to fill the memory capacities of the GPU (11GB). For both representations, the learning rate is initialized to $1 \cdot 10^{-4}$ for FCN-8s and $5 \cdot 10^{-5}$ for FCN-16s and FCN-32s. The learning rate has an exponential decay of 0.9 each 10 epochs. Training has been completed using the PyTorch framework with a single *GeForce RTX 2080 Ti* GPU.

Performances are evaluated for each RADAR representation using the IoU, the Pixel Precision (PP) and the Pixel Recall (PR) for each category. Metrics by category are aggregated using arithmetic and harmonic means. To ensure consistency of the results, all performances are averaged from three trained models initialized with different seeds. Results are presented in Table 4.6.

Models are trained on dense mask annotations and evaluated on both dense mask (top values) and sparse points (bottom values in parentheses) annotations. Sparse points are

more accurate than dense masks, therefore evaluation on this type of annotation provides information on the behaviour of predictions on key points. However, localization should not be evaluated for sparse points using a model trained on dense masks, therefore IoU performances are not reported. The background category cannot be assessed for the sparse points because some of the points should belong to an object but are not annotated *per se*. Thus, arithmetic and harmonic means of sparse points evaluations are computed for only three classes against four for the dense masks.

The baseline shows that meaningful representations are learnt by a popular visual semantic segmentation architecture. These models succeed in detecting and classifying shapes of moving objects in raw RADAR representations even with sparse-point annotations. Performances on RA are not as good as in RD because the angular resolution of the sensor is low, resulting in less precise generated annotations. An extension to improve performances on this representation could be to transform it into Cartesian coordinates as done in [Major *et al.* 2019]. For both representations, results are promising since the temporal dimension of the objects signatures has not yet been taken into account.

4.3.4 Discussions

The semi-automatic algorithm presented in Section 4.3.2 generates precise annotations on raw RADAR data, but it has limitations. Occlusion phenomena are problematic for tracking, since they lead to a disappearance of the object point cloud in the DoA-Doppler representation. An improvement could be to detect such occlusions in the video frames and include them in the tracking pipeline. The clustering in the DoA-Doppler representation is also a difficult task in specific cases. When objects are close to each other with a similar radial velocity, point clouds are difficult to distinguish. Further work on the bandwidth selection and optimisation of this selection could be explored.

The CARRADA dataset provides precise annotations to explore a range of supervised learning tasks. A simple baseline is proposed for semantic segmentation trained on dense mask annotations. It could be extended by using temporal information or both dense mask and sparse points annotations at the same time during training. Current architectures and loss functions could also be optimized for semantic segmentation of sparse ground-truth points. Object detection could be considered by using bounding boxes to detect and classify object signatures. As off-the-shelf object detection algorithms are not adapted to the RADAR data representation and to the unusual size of the provided annotations, further work is required to redesign these methods. By identifying and tracking specific instances of objects, other opportunities are opened. Tracking of sparse points or bounding boxes could also be considered.

4.3.5 Conclusions

The CARRADA dataset contains synchronised video frames, Range-Angle-Doppler tensor and thus, Range-Angle and Range-Doppler raw RADAR representations. RADAR data are annotated with sparse points, bounding boxes and dense masks to localize and categorize the object signatures. A unique identification number is also provided for each instance.

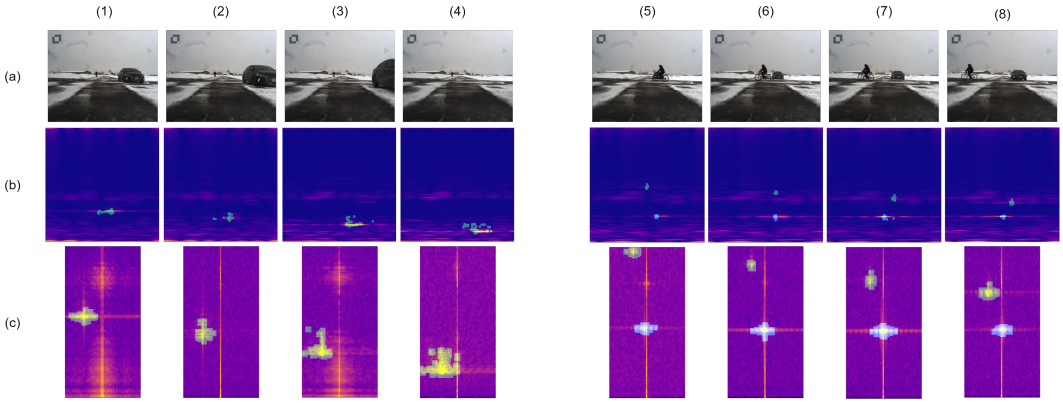


Figure 4.12: **Two scenes from CARRADA dataset, one with a car, the other on with a cyclist and a car**. (a) Video frames provided by the frontal camera showing moving objects in a fixed environment; (b-c) RADAR signals at the same instants in Range-Angle and cropped Range-Doppler representation respectively. (1-4) First sequence; (5-8) Second sequence. Frames in both sequences are selected with an interval of 5 frames. In the first sequence, the segmentation mask corresponds to the annotation of the approaching car in the scene. Range-Doppler data (c)(1) and (c)(3) show that our method is robust to recording noise. In frames (4), the car is still in the radar's field of view but it has disappeared from the camera. In the second sequence, the segmentation masks correspond respectively to the annotations of the moving cyclist (blue) and car (green). The cyclist is moving from right to left in front of the RADAR, its radial velocity is progressively changing from positive to negative. Note that the Range-Angle view illustrated in (b) has been obtained using a maximum aggregation and log-transform in Equation 2.18 while the Range-Doppler view in (c) is computed using Equation 2.19.

Annotations are generated using a semi-supervised algorithm based on visual and physical knowledge. The proposed semi-automatic pipeline generating the annotations is robust to RADAR sensor noise and camera occlusion while being able to track an object in the RADAR data which is not visible in the camera. These results are illustrated in Figure 4.12. This pipeline could be used to annotate any camera-RADAR recordings with similar settings. The dataset, code for the annotation algorithm and code for dataset visualisation are publicly available [4]. Even if the recorded scenes contain only one or two objects moving in a controlled environment, the dataset is unique in its kind by providing raw RADAR data with detection and semantic segmentation annotations. This work also aims to motivate deep learning research applied to RADAR sensor and multi-sensor fusion for scene understanding as proposed in the following chapter.

4.4 Conclusions

This chapter has outlined methods to address the lack of open and annotated RADAR data. We presented a simple simulation of objects' signature in RD representations by consid-

[4] https://github.com/valeoai/carrada_dataset

ering the geometric properties of a moving object in a scene and incorporating different types of noise (Section 4.1). The proposed simulator is too simple to generate realistic samples. As our experiments show, deep neural network architectures are capable of easily over-fitting the simulation.

Secondly, we proposed methods to generate RD representations from camera images using real data recorded in complex urban scenes (Section 4.2). Our proposed AE architecture combined with the "Noise Map" method, reached the best quantitative and qualitative results. However, the results were not convincing enough to further explore the transfer of annotations from the source domain (camera image) to the target domain (RD representation).

Finally, we presented CARRADA, a unique dataset including synchronized camera images and raw RADAR data with object detection and semantic segmentation annotations (Section 4.3). We also proposed a semi-automatic pipeline based on camera images generating annotations on RADAR representations while being robust to object occlusion, narrow camera FoV and RADAR sensor noise, as illustrated in Figure 4.12. The CARRADA dataset and the presented annotation method were presented at the International Conference on Pattern Recognition (ICPR).

The proposed dataset contains simple scenes with only one or two moving objects at the same time. However, it is an interesting source of data for designing deep neural network architectures, while being realistic enough to be generalized to complex urban scenes, as we detail in the next chapter. The annotation pipeline is also limited in distinguishing two objects with similar Doppler while being close together. Furthermore, it requires a human intervention to instantiate the tracking method. This is a preliminary work on automatic RADAR annotation generation, which has been extended in the recent work of [Zhang *et al.* 2021a], as detailed in Section 5.1.4.1.

In the following chapter, we will present methods for RADAR scene understanding. First, we will propose deep neural architectures for multi-view RADAR semantic segmentation with their corresponding loss functions. Then, we will introduce a method for LiDAR and RADAR point cloud fusion to take advantage of both sensors for scene understanding applications.

CHAPTER 5

RADAR scene understanding

Contents

5.1 Multi-view RADAR semantic segmentation 86
5.1.1 Motivations 86
5.1.2 Methods and architectures 86
5.1.3 Experiments on the CARRADA dataset 92
5.1.4 Experiments on complex urban scenes datasets 96
5.1.5 Conclusions and perspectives 100
5.2 Sensor fusion 101
5.2.1 Introduction 101
5.2.2 Method 102
5.2.3 Simulation 106
5.2.4 Application to the nuScenes dataset 106
5.2.5 Discussions and future work 108
5.3 Conclusions 109

The RADAR sensor is recently a source of interest since public datasets have been released as detailed in Section 3.2. RADAR scene understanding is in its infancy, but it provides key information to compensate for weaknesses of the other sensors. The RADAR data presented as a RAD tensor contains signatures of the objects surrounding the car with enough details to distinguish them in the representation. Unlike object detection using bounding boxes, semantic segmentation is appropriate for this task, since the object signatures have extremely variable sizes and may be mixed up due to the sensor's resolution.

Multi-sensor fusion is essential to improve scene understanding by taking advantage of complementary sensor properties. LiDAR and RADAR point clouds are easy to fuse in the Cartesian coordinates around the ego vehicle. Even if the RADAR point cloud is degraded through the signal processing pipeline (see Section 2.2), it offers reflectivity and velocity information useful to localise objects in the dense LiDAR dense representation.

In this chapter, we propose a method for RADAR scene understanding. First, a multi-view approach is proposed for RADAR semantic segmentation in Section 5.1 details neural network architectures and their associated loss functions adapted to the RAD tensor. This section is the second main contribution of this thesis, it is mainly inspired from our article published at the International Conference on Computer Vision (ICCV) [Ouaknine *et al.* 2021]. Section 5.2 introduces a method to fuse and propagate RADAR point cloud properties through the LiDAR point cloud aiming to improve scene understanding tasks.

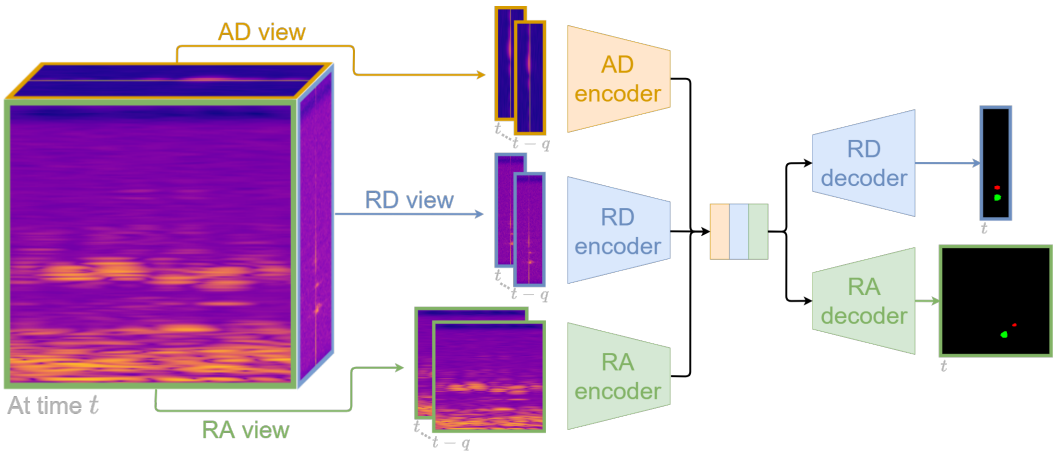


Figure 5.1: **Overview of our multi-view approach to semantic segmentation of RADAR signal**. At a given instant, RADAR signals take the form of a Range-Angle-Doppler tensor. Sequences of $q+1$ 2D views of this data cube are formed and mapped to a common latent space by our proposed multi-view architectures. Two heads with distinct decoders produce a semantic segmentation of the Range-Angle and Range-Doppler views respectively ('background' in black, 'pedestrian' in red and 'cyclist' in green in this example).

5.1 Multi-view RADAR semantic segmentation

5.1.1 Motivations

In this section, an approach to multi-view RADAR semantic segmentation is proposed, illustrated in Figure 5.1, that exploits the entire data while addressing the challenges of its large volume and high level of noise (see Appendix B.1.1). The segmentation is performed on the RD and RA views, which suffices to deduce the localisation and the relative speed of objects. The first contribution is a set of lightweight neural network architectures designed for multi-view semantic segmentation of RADAR signal (see Section 5.1.2.3. The second contribution is a set of loss terms to train models on these tasks while preserving coherence between the multi-view predictions (see Section 5.1.2.4). Experiments in Sections 5.1.3 and 5.1.4 demonstrate that our proposed best model outperforms alternative models, derived either from the semantic segmentation of natural images or from RADAR scene understanding, while requiring significantly fewer parameters. Both our code and trained models are available online[1]. Section 5.1.5 finally concludes and proposes perspectives.

5.1.2 Methods and architectures

The following Sections 5.1.2.1 and 5.1.2.2 briefly present several methods for image segmentation and RADAR scene understanding, to which this work is compared. They are chosen for their performance and their relevance to the RADAR semantic segmentation task. Further details on the architectures are provided in Sections 2.6.3 and 3.4. Except

[1] https://github.com/valeoai/MVRSS

RSS-Net [Kaul *et al.* 2020], these architectures were not originally designed for RADAR semantic segmentation, nor to handle multiple views of a data volume. Consequently, for each of them, two models have been trained independently for RD and RA segmentation respectively. More details are provided in Section 5.1.3.2.

The three proposed architectures for multi-view RADAR semantic segmentation are introduced in Section 5.1.2.3.

5.1.2.1 Image-based competing methods

Long *et al.* [Long *et al.* 2015] propose FCN, consisting of convolutional and down-sampling layers followed by transposed convolutions ("up-convolutions"). The final representations are processed by a 1D convolution with softmax to predict a category for each pixel. Several versions are proposed depending on the feature-map scales used to generate the output. FCN has been used for semantic segmentation of RADAR data in Section 4.3.3, where FCN-8s version achieves the best performance.

The U-Net architecture [Ronneberger *et al.* 2015] is composed of equal-depth down-sampling and up-sampling pathways linked by skip connections. Originally used for medical images, it is especially well suited for small-object segmentation.

The DeepLabv3+ [Chen *et al.* 2018b] is a popular encoder-decoder model for semantic segmentation of natural images. The encoder uses "atrous" separable convolutions which increase the receptive field of the network. The proposed ASPP layer [Chen *et al.* 2018a] combines atrous convolutions with various dilation rates to learn multi-scale features followed by a 1D convolution.

These methods are presented in detail in Section 2.6.3.

5.1.2.2 Radar-based competing methods

Kaul *et al.* [Kaul *et al.* 2020] propose RSS-Net, specialised in RADAR semantic segmentation, in particular for RA representations. Its architecture is similar to DeepLabv3+ with an encoder composed of 8 atrous convolutional layers, an ASPP module and a convolutional decoder with up-sampling. The architecture is designed to reduce the dimension of the feature maps in the encoder, leading to excellent performance for Range-Angle BEV semantic segmentation.

Gao *et al.* [Gao *et al.* 2020] propose the RADAR Multiple-Perspective Neural Network (RAMP-CNN) for object detection in RA representations. They aggregate the RAD tensor into 2D RADAR views which are processed by separate encoder-decoders with 3D convolutional layers. The final RA features are processed by a 3D inception module. RAMP-CNN achieves state of the art performance in localising and classifying multiple objects in RA views.

These methods are presented in detail in Section 3.4.

5.1.2.3 Proposed multi-view methods

We proposed four lightweight neural network architectures. They are specialised in multi-view RADAR semantic segmentation, whose general principle is illustrated in Figure 5.2.

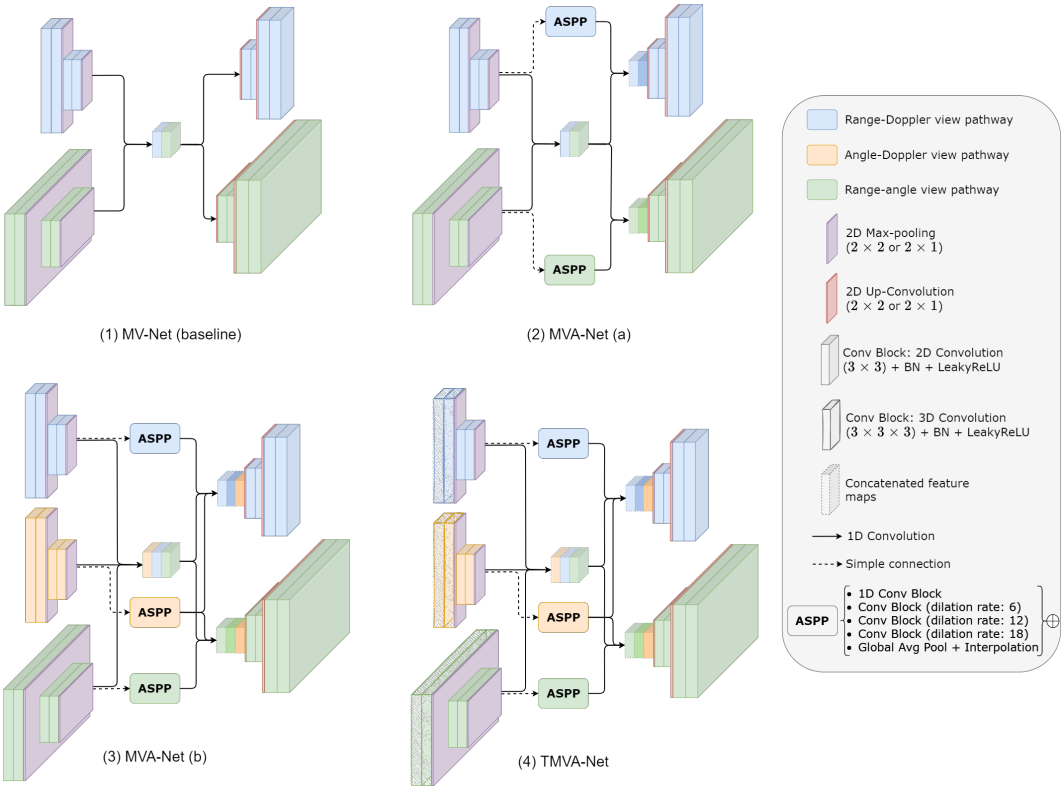


Figure 5.2: **Multi-view architectures for RADAR semantic segmentation**. (1) The multi-view network (MV-Net), considered as a baseline, is composed of two encoders, two decoders and a common latent space. (2) The MVA-Net(a) has an additional Atrous Spatial Pyramidal Pooling module for each view pathway; (3) The MVA-Net(b) has an additional Angle-Doppler pathway; (4) The TMVA-Net architecture is similar to the MVA-Net(b) with 3D convolutions at the top of the encoders exploiting the temporal dimension. The detailed architectures are provided in Appendix B.1.2.

They take a temporal stack of RADAR views as their input and process them with dedicated encoders. The generated feature maps are fused in a shared latent space from which different decoders predict semantic segmentation maps for each output view. Since neither encoders nor decoders share weights, they are kept simple to reduce the size of the total network. Thus, there are specialised parts of the network in each view, and a reasonable number of parameters altogether. The proposed architectures are detailed layer by layer in Appendix B.1.2.

Multi-view network (MV-Net). Firstly, we proposed a baseline (see Figure 5.2 (1)) in the form of a double encoder-decoder architecture that processes stacked RD and RA views and predicts simultaneously the RD and RA semantic segmentation maps. It is important to note that the third view (AD) can be deduced from the RA and RD views, and it is therefore not necessary to segment it in the output. This is the case for all proposed architectures.

Each encoder is composed of two blocks, each one with two sequences of convolution, batch normalisation and activation layers. The two blocks are separated by a max-pooling

operation to down-sample the feature maps along the range axis (the Doppler's resolution being lower, it is kept unchanged). The feature maps from both encoders are down-sampled, processed by a 1D convolution and stacked into a common latent space. Since the input views are stacked according to the time dimension, this linear combination of the feature maps aims to learn temporal correlations. The features in the shared latent space are then processed with 1D convolution layers and used as the input of each decoder. There are two decoders predicting respectively the RD and RA semantic segmentation maps. Each one is composed of two blocks with two sequences of convolution, batch normalisation and activation layers. The up-sampling between the blocks is carried out by up-convolutions. A final 1D convolution performs a combination of the outputs of each decoder to generate K feature maps, where K is the number of classes. A softmax operation is then applied pixel-wise to the K feature maps generating soft masks.

Multi-view network with ASPP modules (MVA-Net). The ASPP module [Chen *et al.* 2018a] used in DeepLabv3 [Chen *et al.* 2018b] allows features to be jointly learned at different scales at a given depth of the network with no need for larger kernels or additional parameters. As shown in RSS-Net [Kaul *et al.* 2020], it is well suited for RADAR semantic segmentation since the objects' signatures can vary considerably. The second architecture, MVA-Net, introduces the use of this ASPP module at the end of each decoder of our MV-Net baseline. The generated multi-scale feature maps are concatenated, processed by a 1D convolution and stacked to the input of each corresponding decoder. Two variants of MVA-Net are proposed: MVA-Net(a), shown in Figure 5.2 (2), consists in a double encoder-decoder architecture with ASPP modules to process and segment RD and RA views; MVA-Net(b), shown in Figure 5.2 (3), has an additional encoding branch learning features from the AD view. Similarly to the other encoding branches, the AD backbone generates features that are stacked in the common latent space. However, the outputs of its ASPP module contain both the angle and Doppler information. Thus the multi-scale feature maps from the AD pathway are stacked to the inputs of both the RD and RA decoders.

Temporal multi-view network with ASPP modules (TMVA-Net). The temporal dimension provides valuable information for RADAR semantic segmentation. It helps in estimating the shape of an object's signature despite high noise levels, and distinguishing objects that are close together with similar velocities. The third architecture, which will show to have the best performances, TMVA-Net in Figure 5.2 (4), extends MVA-Net by explicitly leveraging the temporal dimension. The first block's 2D convolutions are replaced by 3D convolutions in each encoder branch, making it able to learn the spatio-temporal characteristics with limited increase in the number of parameters. Since 3D convolutions require a large number of parameters, full 3D-convolutional encoders, such as in [Gao *et al.* 2020], have not been retained. Hence, TMVA-Net is composed of three encoders with 3D and 2D convolutions, one for each input view. Each one of them has a dedicated ASPP module. The feature maps generated from each encoding backbone are stacked into a shared latent space. From there, two decoders segment respectively the RD and RA views. They take as input the stacked features from the processed latent space and the multi-scale feature maps from the dedicated ASPP modules.

5.1.2.4 Losses

In what follows, the following generic notation are used: $f_\theta(\mathbf{x}) = \mathbf{p}$ for a segmentation model with parameters θ, input $\mathbf{x}$ and output $\mathbf{p}$. Training f_θ amounts to minimising w.r.t. θ a suitable loss function, given training examples $\mathbf{x}$ with ground truth $\mathbf{y}$. The architectures presented in Sections 5.1.2.1 and 5.1.2.2 take single-view inputs stacked in the temporal dimension and predict, for each target view, a soft segmentation mask with class "probabilities" for each bin. For instance, the output of a model processing only the RA view is $f_\theta(\mathbf{x}^{\text{RA}}) = \mathbf{p}^{\text{RA}} \in [0,1]^{B_\text{R} \times B_\text{A} \times K}$, if $B_\text{R} \times B_\text{A}$ is the size of the view and K the number of classes. Our architectures, detailed in Section 5.1.2.3, take instead multi-view inputs: either $\mathbf{x} = (\mathbf{x}^{\text{RD}}, \mathbf{x}^{\text{RA}})$ or $\mathbf{x} = (\mathbf{x}^{\text{RD}}, \mathbf{x}^{\text{AD}}, \mathbf{x}^{\text{RA}})$. In both cases, their goal is to predict soft masks $\mathbf{p} = (\mathbf{p}^{\text{RD}}, \mathbf{p}^{\text{RA}})$ for both RD and RA views.

The following section details the loss functions applied to each segmented view to train our proposed architectures. A "coherence" loss is also introduced to enforce consistency between the predictions over the two views of the scene. Finally a combination of these loss terms is proposed.

Weighted Cross Entropy. Semantic segmentation models that predict a score for each class at each pixel are usually trained by minimising a CE loss function. This loss is not ideal for unbalanced segmentation tasks such as the RADAR semantic segmentation, since the optimisation process tends to focus on the classes that are most represented. In the present case, background and speckle noise dominate, in comparison to the signatures of the objects we wish to detect. In the same manner as RSS-Net [Kaul *et al.* 2020], a weighted Cross-Entropy (wCE) loss is employed to tackle this issue.

Given a training example $\mathbf{x}$, let $\mathbf{y} \in \{0,1\}^{B_\text{M} \times B_\text{N} \times K}$ be its one-hot ground truth and $f_\theta(\mathbf{x}) = \mathbf{p} \in [0,1]^{B_\text{M} \times B_\text{N} \times K}$ the associated prediction where $B_\text{M} \times B_\text{N}$ is the size of the view $\mathbf{x}$. The wCE loss function is defined as:

$$\mathcal{L}_{\text{wCE}}(\mathbf{y}, \mathbf{p}) = -\frac{1}{K} \sum_{k=1}^{K} w_k \sum_{(m,n) \in \Omega} \mathbf{y}[m,n,k] \log \mathbf{p}[m,n,k], \tag{5.1}$$

where $\Omega = [\![1, B_\text{M}]\!] \times [\![1, B_\text{N}]\!]$, and w_k's are normalized positive weights. Weight w_k is inversely proportional to the frequency of class k in the training set, that is $w_k \propto \left(\sum_{\mathbf{y}} \sum_{(m,n) \in \Omega} \mathbf{y}[m,n,k] \right)^{-1}$. The fewer the bins with ground-truth class k, the larger w_k becomes.

Soft Dice. Object signatures in RADAR representations often correspond to small regions. This is a well known issue in medical image segmentation, where the Dice metric (detailed in Section 5.1.3.1) is usually reformulated in a function called Dice loss, ranging between 0 and 1. In their work, [Milletari *et al.* 2016] have proposed the Soft Dice (SDice) loss defined as:

$$\mathcal{L}_{\text{SDice}} = \frac{1}{K} \sum_{k=1}^{K} \left[1 - \frac{2 \sum_{(m,n)} \mathbf{y}[m,n,k] \mathbf{p}[m,n,k]}{\sum_{(m,n)} \mathbf{y}^2[m,n,k] + \mathbf{p}^2[m,n,k]} \right], \tag{5.2}$$

where $(m,n) \in \Omega$, as in Equation 5.1. This formulation has proved useful for 2D and 3D

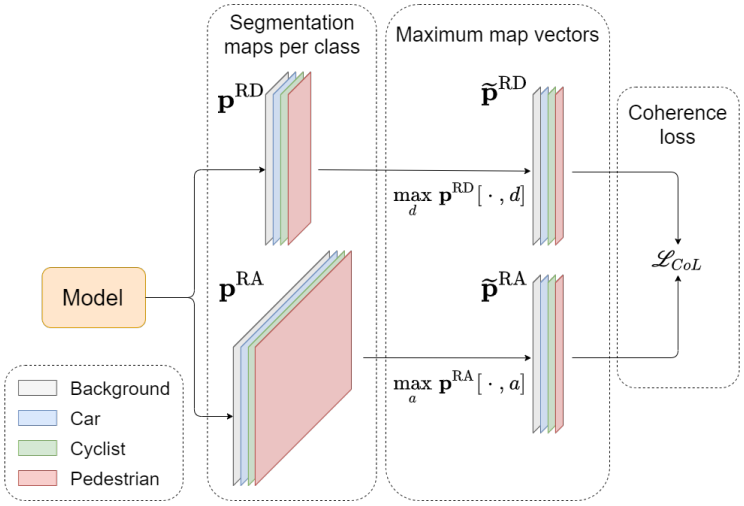


Figure 5.3: **Computation of the coherence loss**. The segmentation maps $\mathbf{p}^{\mathrm{RD}}$ and $\mathbf{p}^{\mathrm{RA}}$ of the two views are aggregated by max pooling along the axis that they do not share (either the Doppler or the angle). The coherence loss is the Mean Squared Error between the two resulting vectors $\tilde{\mathbf{p}}^{\mathrm{RD}}$ and $\tilde{\mathbf{p}}^{\mathrm{RA}}$.

medical image semantic segmentation, including for small objects.

Coherence. The objective of multi-view RADAR semantic segmentation is to simultaneously segment several views of the aggregated RAD tensor. The objects we wish to detect are observed in the different RADAR views, thus it is clear that a certain coherence must be maintained between the segmented views. For example, one view should not represent a pedestrian, while another represents a cyclist. A Coherence Loss (CoL) is introduced to preserve a consistency between the predictions of the model. The procedure to calculate this loss is illustrated in Figure 5.3.

Let $(\mathbf{p}^{\mathrm{RD}}, \mathbf{p}^{\mathrm{RA}})$ be the segmentation maps predicted by the model f_θ after the softmax operation. These two maps are aggregated by applying a $\max(.)$ operator along the axis that they do not share (either the Doppler or the angle). The two resulting maps of same size, denoted $\tilde{\mathbf{p}}^{\mathrm{RD}}$ and $\tilde{\mathbf{p}}^{\mathrm{RA}}$, contain the highest probability of each range bin for each class. In other words, they indicate if the model predicts a high probability to observe a category at a given distance. The coherence loss is the MSE between these maximum range probability vectors. It encourages the network to predict high probability values at the same distance and in the same class for both views. The CoL, in the interval $[0, 1]$, is defined as:

$$\mathcal{L}_{\mathrm{CoL}}(\mathbf{p}^{\mathrm{RD}}, \mathbf{p}^{\mathrm{RA}}) = \frac{1}{B_{\mathrm{R}} \cdot K} \left\| \tilde{\mathbf{p}}^{\mathrm{RD}} - \tilde{\mathbf{p}}^{\mathrm{RA}} \right\|_{\mathrm{F}}^2, \tag{5.3}$$

where $\| \cdot \|_{\mathrm{F}}$ denotes the Frobenius norm, B_{R} the number of range bins and K the number of classes.

Combination of losses. The CE loss is specialised in pixel-wise classification and does not consider spatial correlations between the predictions. The SDice is particularly effec-

tive for shape segmentation, but it is difficult to optimise as a single loss function due to its gradient formulation. Finally, the CoL is useful where neither the CE nor the SDice is able to leverage a coherence between the prediction of the RD and the RA views. To combine the different strengths of these losses, the following final loss is proposed to train multi-view architectures:

$$\mathcal{L} = \lambda_{\text{wCE}}(\mathcal{L}^{\text{RD}}_{\text{wCE}} + \mathcal{L}^{\text{RA}}_{\text{wCE}}) + \lambda_{\text{SDice}}(\mathcal{L}^{\text{RD}}_{\text{SDice}} + \mathcal{L}^{\text{RA}}_{\text{SDice}}) + \lambda_{\text{CoL}}\mathcal{L}_{\text{CoL}}, \tag{5.4}$$

where λ_{wCE}, λ_{SDice} and λ_{CoL} are weighting factors set empirically.

5.1.3 Experiments on the CARRADA dataset

This section presents the experimental evaluation of our proposed models on the CARRADA dataset described in Section 4.3. The dataset and the evaluation metrics are briefly described. Modification made to the competing methods are then explained. Finally, details are provided concerning the experiments and analyse their results quantitatively and qualitatively.

5.1.3.1 Dataset and evaluation metrics

Dataset. The CARRADA dataset contains synchronised camera and automotive RADAR recordings with 30 sequences of various scenarios with one or two moving objects. The RADAR views are annotated using a semi-automatic pipeline (see Section 4.3.2). This is the only publicly-available dataset providing RAD tensors and dense semantic segmentation annotation for both RD and RA views. The objects are separated into four categories: *pedestrian*, *cyclist*, *car* and *background*. The provided RAD tensors have dimensions $B_{\text{R}} \times B_{\text{A}} \times B_{\text{D}} = 256 \times 256 \times 64$. Additional details on the dataset are provided in Section 4.3.1. The experiments presented in Section 5.1.3.3 use the proposed dataset splits (see Figure 4.6, denoted CARRADA-Train, CARRADA-Val and CARRADA-Test.

Evaluation metrics. A classic performance metric in semantic segmentation is the IoU This metric has been introduced in Section 2.6.2. From the perspective of a single object in a given scene, the IoU measures how well and how completely it is segmented. Averaging this metric over all classes yields the mIoU score. Another related, yet slightly different metric, is provided by the Dice score: For a given class and with same notations as above, it is defined as $\frac{2|A \cap B|}{|A|+|B|}$. For global performance, it is averaged over all classes into the mean Dice (mDice). Seeing segmentation as a local 1-*vs*.-all classification problem for each class, the Dice amounts to the harmonic mean of the precision and recall (a.k.a. F1 score). The IoU and Dice metrics are considered as complementary; Both of them are reported in our experiments.

5.1.3.2 Implementation of competing methods

This section describes the architectures used for comparisons. For each method, one model is trained specifically for single-view semantic segmentation of either RD or RA. Details concerning pre-processing procedures are provided in the Appendix B.1.3.

The non-radar-based architectures have been used "as is": The FCN-8s architecture is based on a VGG16 [Simonyan & Zisserman 2015] backbone; DeepLabv3+ uses a ResNet-101 [He *et al.* 2016]; The U-Net architecture is identical to the one in [Ronneberger *et al.* 2015].

In the experiments with RSS-Net, the number of down-sampling layers in the encoding part has been reduced to be trained with lower resolution inputs.

The RAMP-CNN architecture dedicated to the RD view has been adapted with two major changes. Firstly, the fusion module has been modified to aggregate and duplicate the feature maps to suit the RD space. Secondly, the size of the output feature maps has been reduced on the Doppler axis using an additional convolutional layer with 3×3 filters and a stride factor of 4. For both RD and RA segmentation tasks, an additional 1D convolutional layer with a softmax operation processes the last feature maps to predict segmentation maps.

5.1.3.3 Training and results

Training procedures. Methods presented in Section 5.1.2 are trained using CARRADA-Train and CARRADA-Val splits and tested on the CARRADA-Test. At each timestamp of a RADAR sequence, the provided RAD tensor is processed according to the method presented in Section 2.2.

For each frame, q past frames are also considered for both training and testing phases: The views from $t-q$ to t are stacked into the time-t input (see Figure 5.1). For the methods that do not explicitly process the time dimension, $q = 2$ for a total input sequence length of 3. Time-based methods using 3D convolutions have specific sequence lengths: $q = 8$ for RAMP-CNN and $q = 4$ for TMVA-Net.

The competing architectures have been trained with the CE loss, except for the RSS-Net, which is trained with a wCE using the formulation in [Kaul *et al.* 2020]. Our proposed methods are trained using the combination of loss terms detailed in Section 5.1.2.4. Our proposed formulation of the wCE loss has been used (Section 5.1.2.4) with weights computed on CARRADA-Train.

All the training procedures use the Adam optimiser [Kingma 2015] with the recommended parameters ($\beta_1 = 0.9$, $\beta_2 = 0.999$ and $\varepsilon = 10^{-8}$). Since each method has its own set of hyper-parameters, further details are provided in the Appendix B.1.3, namely batch sizes, learning rates, learning rate decays, numbers of epochs and corresponding pre-processing steps for each one of them. Training was performed using the PyTorch framework with a single GeForce RTX 2080 Ti graphic card.

Quantitative results. The performance for both RD and RA semantic segmentation tasks on CARRADA-Test are shown in Table 5.1. Our proposed TMVA-Net achieves the best scores for both mDice and mIoU metrics and for both segmentation tasks. Moreover, the proposed methods are the only ones to perform both tasks simultaneously. TMVA-Net also provides the best trade-off between performance and number of parameters for both tasks, as illustrated in Figure 5.4 with mDice metric (similar plots with mIoU metric are presented in Appendix B.1.4). A study of performance variability considering four trained models for each method leads to the same conclusion as detailed in Appendix B.1.5. Note that the

View	Method	# Param. (M)	IoU (%)					Dice (%)				
			Bkg.	Ped.	Cyclist	Car	**mIoU**	Back.	Ped.	Cyclist	Car	**mDice**
RD	FCN-8s [Long *et al.* 2015]	134.3	99.7	47.7	18.7	52.9	54.7	99.8	24.8	16.5	26.9	66.3
	U-Net [Ronneberger *et al.* 2015]	17.3	99.7	51.0	**33.4**	37.7	55.4	99.8	67.5	**50.0**	54.7	68.0
	DeepLabv3+ [Chen *et al.* 2018b]	59.3	99.7	43.2	11.2	49.2	50.8	99.9	60.3	20.2	66.0	61.6
	RSS-Net	10.1	99.3	0.1	4.1	25.0	32.1	99.7	0.2	7.9	40.0	36.9
	RAMP-CNN	106.4	99.7	48.8	23.2	54.7	56.6	99.9	65.6	37.7	70.8	68.5
	MV-Net (ours-baseline)	2.4*	98.0	0.0	3.8	14.1	29.0	99.0	0.0	7.3	24.8	32.8
	MVA-Net (a) (ours)	3.6*	99.7	26.5	20.7	48.8	48.9	99.8	41.9	34.3	65.6	60.4
	MVA-Net (b) (ours)	4.8*	99.7	30.2	22.0	**59.6**	52.9	99.9	46.4	36.1	**74.7**	64.3
	TMVA-Net (ours)	5.6*	99.7	**52.6**	29.0	53.4	**58.7**	99.8	**68.9**	45.0	69.6	**70.9**
RA	FCN-8s [Long *et al.* 2015]	134.3	99.8	14.8	0.0	23.3	34.5	99.9	25.8	0.0	37.8	40.9
	U-Net [Ronneberger *et al.* 2015]	17.3	99.8	22.4	**8.8**	0.0	32.8	99.9	36.6	**16.1**	0.0	38.2
	DeepLabv3+ [Chen *et al.* 2018b]	59.3	99.9	3.4	5.9	21.8	32.7	99.9	6.5	11.1	35.7	38.3
	RSS-Net	10.1	99.5	7.3	5.6	15.8	32.1	99.8	13.7	10.5	27.4	37.8
	RAMP-CNN	106.4	99.8	1.7	2.6	7.2	27.9	99.9	3.4	5.1	13.5	30.5
	MV-Net (ours-baseline)	2.4*	99.8	0.1	1.1	6.2	26.8	99.0	0.0	7.3	24.8	28.5
	MVA-Net (a) (ours)	3.6*	99.0	0.0	5.9	7.5	28.1	99.5	0.0	11.1	14.0	31.1
	MVA-Net (b) (ours)	4.8*	99.8	6.6	7.4	**32.9**	36.7	99.9	12.5	13.8	**49.5**	43.9
	TMVA-Net (ours)	5.6*	99.8	**26.0**	8.6	30.7	**41.3**	99.9	**41.3**	15.9	47.0	**51.0**

Table 5.1: **Semantic segmentation performance on the CARRADA-Test dataset for Range-Doppler and Range-Angle views**. The number of trainable parameters (in millions) for each method corresponds to a single view-segmentation model; Two such models, one for each view, are required for all methods but ours. In contrast, the number of parameters reported for our methods ('*') corresponds to a single model that segments both RD and RA views. The RSS-Net and RAMP-CNN methods have been modified to be trained on both tasks (see Section 5.1.3.2). Performances are evaluated with the Intersection over Union and the Dice score per class, and their averages, mIoU and mDice, over the four classes. The best scores are in red and bold type, the second best in blue and underlined.

number of parameters reported for each method in Table 5.1, in Figure 5.4 and in Appendix B.1.4 corresponds to a single trained model, while competing methods require two independent models (hence twice more parameters) to perform both RD and RA segmentation tasks.

Ablation Studies. Table 5.2 reports the performance of the four architectures which have been introduced in Section 5.1.2.3. It shows that the additional ASPP modules in MVA-Net(a) boosts the performance relative to MV-Net for both RD and RA segmentation. The performance is further improved with MVA-Net(b) by the additional encoder that extracts features from the AD view and provides relevant information to separate object signatures. Finally, TMVA-Net is the most effective regardless of the metric, thanks to its ability to learn spatio-temporal features with 3D convolutions. The temporal dimension indeed helps distinguish between objects and the speckle noise, and to categorise them according to the shape variations.

Table 5.3 accesses the importance of the different losses on the performance of TMVA-

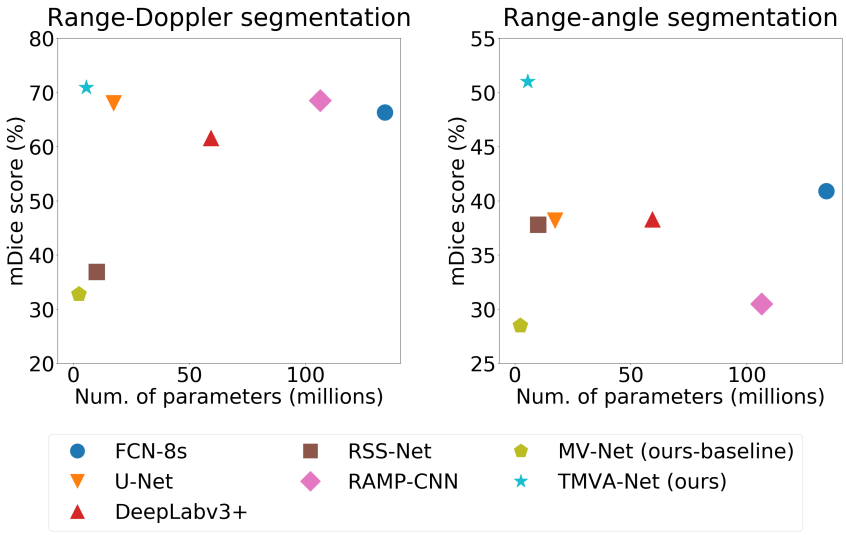


Figure 5.4: **Performance-*vs*.-complexity plots for all methods in Range-Doppler and Range-Angle tasks**. Performance is assessed by mean Dice (%) and complexity by the number of parameters (in millions) *for a single task*. Top-left models are the best performing and the lightest. Only our proposed models, MV-Net and TMVA-Net, are able to segment both views simultaneously. For all the other methods, two distinct models must be trained to address both tasks, which doubles the number of actual parameters.

Net. The best combination of two loss terms is wCE+SDice for both tasks. The performance is further improved on the RA segmentation task by adding our proposed CoL term, while slightly reduced on RD views. This loss improves the coherence between the tasks by better detecting objects in the RA views as discussed in the following section. A loss ablation study considering the performance variability is proposed in Appendix B.1.6. Four models have been trained for each combination of loss functions leading to the same conclusion.

Qualitative results. Figure 5.5 shows qualitative results of each method on a scene from CARRADA-Test. The results of TMVA-Net (i-j) display well segmented RD views in terms of localisation and classification. Only TMVA-Net with CoL (j) is able to localise and classify both objects in the RD and RA views. The enforcement of the coherence of predictions across views succeeds in correctly classifying the same objects in the two views. This is not the case for TMVA-Net without CoL, as illustrated in the example (i), where the model predicts a cyclist instead of a pedestrian in the RA view. Moreover, the coherence loss also helps to discover new objects: In (i), TMVA-Net predicts a single object in the RA view, while in (j), it localises and classifies both objects well with the help of CoL. Additional qualitative results leading to the same conclusions are proposed in Appendix B.1.7.

View	Method	# Param.	mIoU	mDice
RD	MV-Net (baseline)	2.4M	29.0	32.8
	MVA-Net (a)	3.6M	48.9	60.4
	MVA-Net (b)	4.8M	52.9	64.3
	TMVA-Net	5.6M	**59.3**	**71.5**
RA	MV-Net (baseline)	2.4M	26.8	28.5
	MVA-Net (a)	3.6M	28.1	31.1
	MVA-Net (b)	4.8M	36.7	43.9
	TMVA-Net	5.6M	**40.1**	**49.3**

Table 5.2: **Ablation study of our proposed architectures**. Each architecture has been trained using the wCE+SDice combination loss. TMVA-Net delivers the best performances under both mIoU and mDice metrics and for both RD and RA views.

	RD view		RA view	
Loss	mIoU	mDice	mIoU	mDice
CE	56.1	67.8	39.1	48.3
SDice	58.5	70.3	37.1	44.8
wCE	51.1	62.8	34.3	41.1
CE+SDice	45.2	54.0	38.8	46.9
wCE+SDice	**59.3**	**71.5**	40.1	49.3
wCE+SDice+CoL	58.7	70.9	**41.3**	**51.0**

Table 5.3: **Ablation study of the combination of losses**. Each individual or combination of loss(es) is used to train a TMVA-Net model. Our proposed combination (wCE+SDice+CoL) reaches the best mIoU and mDice for the RA view and the second best scores for the RD view.

5.1.4 Experiments on complex urban scenes datasets

This section presents additional experiments in complex urban scenes highlighting the suitability of our proposed TMVA-Net architecture, trained with its corresponding loss function. Quantitative and qualitative results are presented using the RADDet dataset. A qualitative evaluation is also proposed using an in-house dataset without annotations.

5.1.4.1 RADDet dataset

In their work, [Zhang *et al.* 2021a] have proposed a dataset with synchronised RADAR and stereo cameras, a method to annotate the RADAR data and a neural network architecture for RADAR object detection. To the best for our knowledge, it is the only automotive RADAR dataset providing annotated views of the RAD tensor in complex urban scenes.

Dataset. The dataset consists in 10 158 frames of synchronised RADAR and stereo cameras mounted on a stationary car. The scenarios consist in complex urban scenes (crossroads, sidewalks, crowded roads) recorded in Canada. The authors have used a *Texas Instruments* AWR1843-BOOST [2] RADAR sensor as in the CARRADA dataset (see Section 4.3.1). The authors have publicly released RAD tensors of size $B_{\text{R}} \times B_{\text{A}} \times B_{\text{D}} = 256 \times 256 \times 64$ for each frame. They have proposed an annotation method to detect dynamic road users on RADAR representations similarly to the method presented in Section 4.11. First, a Mask R-CNN [He *et al.* 2017] segments objects in a single camera image. A Semi-Global Block Matching method [Hirschmuller 2008] estimates the image depth with stereo cameras. The Cartesian coordinates of each object are then deduced using the calibration matrix of the cameras.

A CFAR algorithm [Rohling 1983] is applied on the RD and RA views of the RAD tensor to extract a point cloud of high intensity values. The point cloud of each RADAR view is is then clustered with a DB-SCAN algorithm [Ester *et al.* 1996]. The object instances

[2] www.ti.com

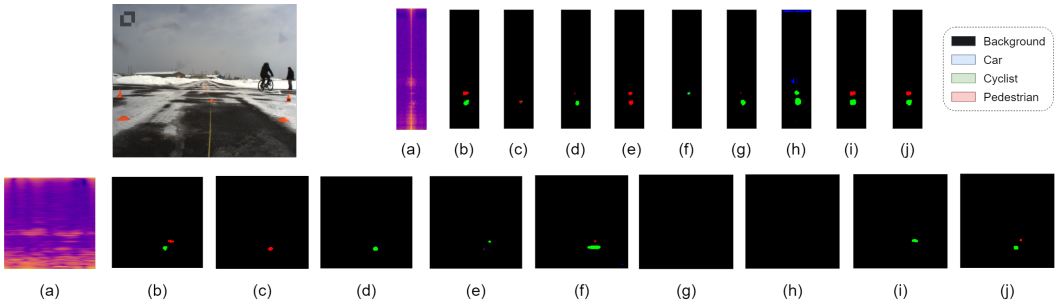


Figure 5.5: **Qualitative results on a test scene of CARRADA**. (*Top*) camera image of the scene and results of the Range-Doppler segmentation; (*Bottom*) Results of the Range-Angle Segmentation. (a) RADAR view signal, (b) ground-truth mask, (c) FCN8s, (d) U-Net, (e) DeepLabv3+, (f) RSS-Net, (g) RAMP-CNN, (h) MV-Net (our baseline w/ wCE+SDice loss), (i) TMVA-Net (ours, w/ wCE+SDice loss), (j) TMVA-Net (ours, w/ wCE+SDice+CoL loss).

from the stereo cameras are projected to a RADAR BEV representation using a projection matrix. The objects are associated to the corresponding clustered point cloud. Finally, the bounding box annotations are deduced from the RADAR point cloud associated to each object.

The annotation pipeline proposed by [Zhang *et al.* 2021a] extends our proposed method presented in Section 4.3.2 by using the depth estimated by stereo cameras. It helps to reduce the projection error from the camera to the Cartesian coordinates. However, their method has a major limitation: it relies on the camera images at each timestamp. It is therefore affected by any ambiguity of camera projection or occlusion problems. A future work could combine both methods to better instantiate the annotation from the camera images and track it in the RADAR sequences. In the RADDet dataset, the selected $10,158$ frames correspond to robust annotations.

The dataset is composed of six unbalanced categories: person, bicycle, car, motorcycle, bus and truck. In the following experiments, classes are grouped to compensate the balance between the classes. Using only three classes as in the CARRADA dataset helps to train and evaluate the architectures proposed in Section 5.1.2.3; and competing methods in Sections 5.1.2.1 and 5.1.2.2. The bicycle and motorcycle categories have been merged to create the two-wheeler class, while the car, bus and truck categories have been reunited in the vehicle class. In their work, [Zhang *et al.* 2021a] neither considered the temporal dimension to build the dataset nor to train their proposed object detection method. The RADDet dataset has been manually rearranged in sequences to suit the training settings detailed in Section 5.1.3.3. The distributions of the grouped categories considering the proposed sequentially splitted RADDet dataset are illustrated in Figure 5.6 (a). The distributions of the proposed training, validation and test splits of the sequential RADDet dataset are illustrated in Figure 5.6 (b).

The RADDet dataset contains bounding box annotations for each object in the RAD tensor and thus, in the RD and RA views. In order to perform RADAR semantic segmenta-

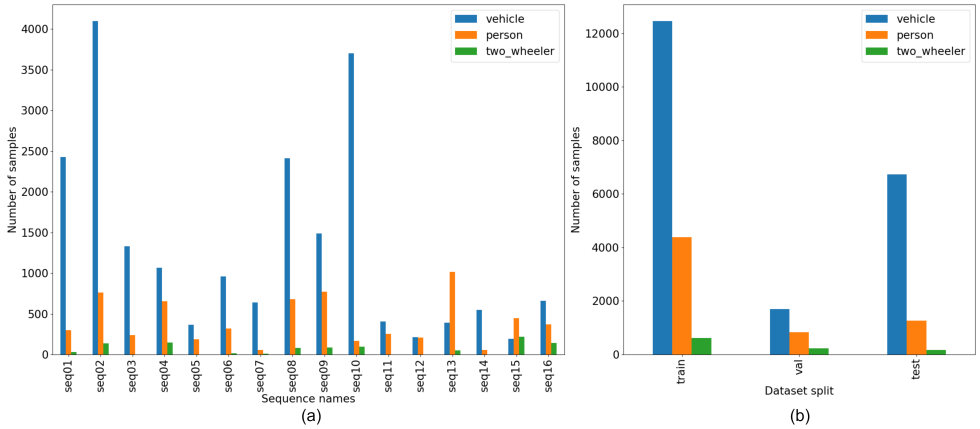


Figure 5.6: **RADDet dataset distributions**. (a) Distribution of each category by sequence; (b) Distribution of each category by proposed dataset split (training, validation and test).

tion, the entire bounding box is considered as a dense segmentation annotation as illustrated in Figure 5.7. There are multiple drawbacks due to this hypothesis: the annotations do not correspond to the object signature in the RADAR representation (it is not rectangular), the annotation contains speckle noise and the model will be penalized if it does not succeed to estimate a rectangle even if the mask covers the signature. However, the CARRADA and the RADDet datasets are the only publicly avaiable datasets providing RAD tensors with object-wise annotations [3]. RADAR semantic segmentation experiments applied to the RADDet dataset are therefore considered as relevant to evaluate our proposed methods in complex urban scenes.

Experiments. The same training procedures as detailed in Section 5.1.3.3 have been followed. The RAD tensors of the RADDet dataset have been aggregated in RADAR views according to Equation 2.19. The methods presented in Sections 5.1.2.1, 5.1.2.2 and 5.1.2.3 have been trained according to their corresponding loss function and set of hyper-parameters detailed in Appendix B.1.3, namely batch sizes, learning rates, learning rate decays, numbers of epochs and corresponding pre-processing steps.

Results. Quantitative results on the RADDet-Test dataset are presented in Table 5.4. Our TMVA-Net architecture, trained with our proposed combination of loss functions (wCE+SDice+CoL), performs the best according to both mIoU and mDice metrics on the RA view segmentation, and to the mDice on the RD view segmentation. Performances according to the mIoU metric on the RD view segmentation are similar than the U-Net method [Ronneberger *et al.* 2015]. However, our proposed method are the only one to perform both task simultaneously. As mentioned in Section 5.1.3.3, TMVA-Net still provides the best trade-off between performance on the RADDet dataset and number of parameters for both tasks.

[3] Note that the methods presented in Sections 5.1.2.1, 5.1.2.2 and 5.1.2.3 are not able to scale with HD RADAR regarding the size of the RAD tensor and the computational cost to estimate it. The following experiments can not be applied to the RADIal dataset presented in Chapter 6.

Method	# Params. (M)	RD view		RA view	
		mIoU	mDice	mIoU	mDice
FCN-8s [Long *et al.* 2015]	134.3	51.3	58.5	31.4	36.0
U-Net [Ronneberger *et al.* 2015]	7.3	49.6	57.1	<u>41.8</u>	<u>49.3</u>
DeepLabv3+ [Chen *et al.* 2018a]	59.3	49.0	55.8	37.9	43.6
RSS-Net	10.1	**54.1**	<u>60.9</u>	39.1	43.8
RAMP-CNN	106.4	44.8	50.6	39.2	45.2
MV-Net	2.4*	41.3	47.8	32.6	37.8
TMVA-Net	5.6*	<u>54.0</u>	**61.8**	**44.2**	**51.4**

Table 5.4: **Semantic segmentation performance on the RADDet-Test dataset for Range-Doppler and Range-Angle views**. The number of trainable parameters (in millions) for each method corresponds to a single view-segmentation model; Two such models, one for each view, are required for all methods but ours. In contrast, the number of parameters reported for our methods ('*') corresponds to a single model that segments both RD and RA views. The RSS-Net and RAMP-CNN methods have been modified to be trained on both tasks (see Section 5.1.3.2). Performances are evaluated with the IoU and the Dice score per class, and their averages, mIoU and mDice, over the four classes. The best scores are bold type, the second best are underlined.

Figure 5.7 shows qualitative results of each method trained on the RADDet-Train and RADDet-Validation datasets, and tested on a scene from RADDet-Test. Once again, the TMVA-Net trained with CoL (i) is the only method to correctly segment two vehicles and a pedestrian in both RD and RA views. The spatial coherence is enforced between the two views enabling the detection of the pedestrian in the RA view. Qualitative results on additional urban scenes are provided in Appendix B.1.8. These results leads to the same conclusions, our proposed method succeeds to segment multiple objects with a spatial coherence consistency outperforming competing methods while performing both tasks simultaneously.

5.1.4.2 In-house dataset

Dataset. For qualitative evaluation only, an in-house dataset has been employed. A few sequences have been recorded by Valeo team members in cities in Canada with a stationary car. The dataset is composed of complex urban scenes with synchronised camera and RADAR data. It has not been released publicly. For these sequences, the RAD tensors have the same dimensions as in CARRADA while the resolution in range is divided by two. The RADAR views are not annotated, which does not allow quantitative evaluation.

Experiments. The RADAR views of this dataset are not annotated and can not be used for training. The methods presented in Sections 5.1.2.1, 5.1.2.2 and 5.1.2.3 have been trained on the CARRADA-Train dataset and tested on the unannotated views of the in-house dataset. The objective is to qualitatively assess the generalisation capacity in complex urban scenes of our proposed methods trained on a simple and controlled environment.

Qualitative results. Qualitative results on these additional urban scenes are provided in Appendix B.1.9. They show that our proposed methods unlike others can generalise well

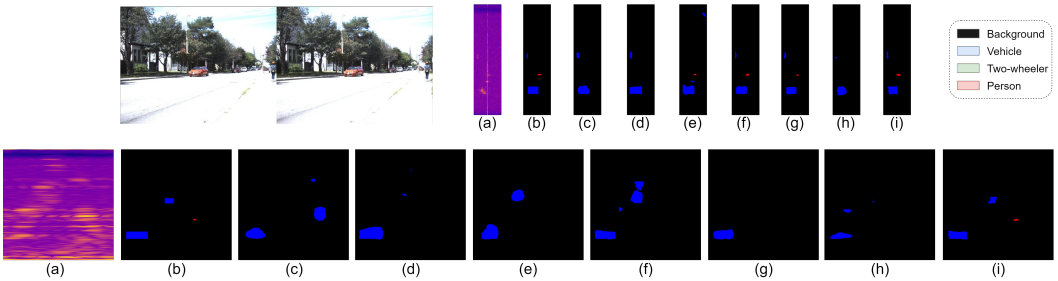


Figure 5.7: **Qualitative results on a test scene of RADDet**. (*Top*) camera image of the scene and results of the Range-Doppler segmentation; (*Bottom*) Results of the Range-Angle Segmentation. (a) RADAR view signal, (b) ground-truth mask, (c) FCN8s, (d) U-Net, (e) DeepLabv3+, (f) RSS-Net, (g) RAMP-CNN, (h) MV-Net (our baseline w/ wCE+SDice+CoL loss), (i) TMVA-Net (ours, w/ wCE+SDice+CoL loss).

on RD and RA views. Indeed, TMVA-Net succeeds in learning object signatures on the CARRADA dataset and recognizing them in a different environment. It also succeeds in detecting and classifying several objects in the scenes, although it has been trained to segment a maximum of two objects at a time.

5.1.5 Conclusions and perspectives

In this section, lightweight architectures are proposed for multi-view RADAR semantic segmentation and a combination of loss terms to train them. Our proposed methods localise and delineate objects in the RADAR scene while simultaneously determining their relative velocity. Experiments show that both the information from the RAD RADAR tensor and from its temporal evolution are important for these tasks. The proposed methods significantly outperform competing architectures specialised either in natural image semantic segmentation or in RADAR scene understanding using the CARRADA dataset. The experiments conducted on the RADDet dataset support these conclusions in complex urban scenes with multiple objects to detect. Preliminary experiments on the in-house dataset also show qualitatively that our method, trained on CARRADA only, generalizes better to new complex urban scenes without fine-tuning.

Our future investigations will focus on improving the segmentation of cyclists and pedestrians, which remain difficult to distinguish. Exploiting RADAR properties could be interesting to improve both RAD tensor aggregation and class-specific data augmentation methods for the benefit of learning algorithms. We note that experiments considering the entire RAD tensor (with or without the temporal information) as input of a neural network architecture were unsuccessful due to the large amount of noise. As a first step, we are exploring aggregation methods applied to the RAD to better separate the object signature distributions while reducing the variance of the speckle noise in the RADAR data. The selection of specific slices per view could also be considered with a measure of signal disparity. We are also trying to reformulate the Coherence Loss by integrating the ground truth, or by back-propagating the information in specific branches of the network.

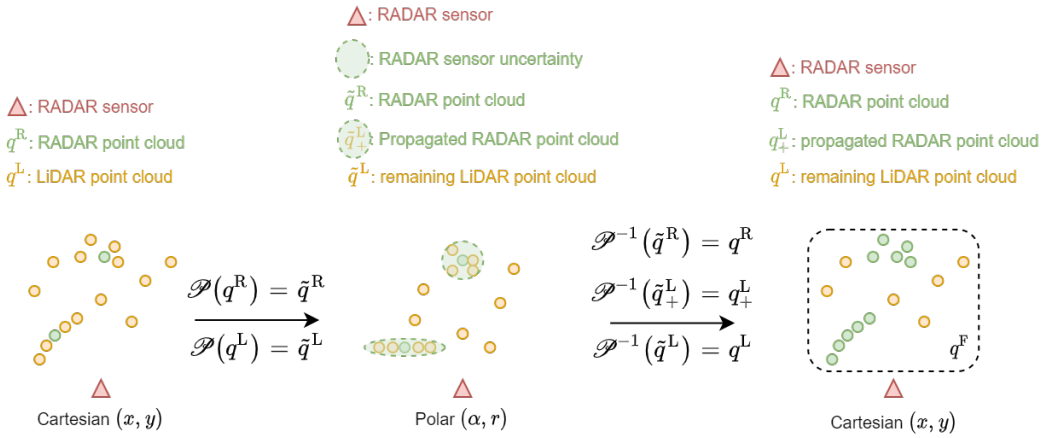


Figure 5.8: **Overview of our propagation and fusion approach for RADAR and LiDAR point clouds**. The RADAR ($\mathbf{q}^{\mathrm{R}}$) and LiDAR ($\mathbf{q}^{\mathrm{L}}$) point clouds in Bird's Eye View Cartesian coordinates are projected in polar coordinates (respectively $\tilde{\mathbf{q}}^{\mathrm{R}}$ and $\tilde{\mathbf{q}}^{\mathrm{L}}$) with $\mathcal{P}(.)$ (see Eq. 5.5). LiDAR points are associated to each RADAR point according to the sensor uncertainties illustrated with green ellipses ($\mathbf{q}^{\mathrm{L}}_{+}$). The Doppler and reflectivities information are shared between the two point clouds in the uncertainty areas. They are projected back in Cartesian coordinates and combined to create the fused point cloud ($\mathbf{q}^{\mathrm{F}}$).

The RADAR sensor has benefits, *e.g.* it provides the Doppler information and it is robust to adverse weather conditions. It has also limitations, *e.g.* low angular resolution and ghost reflections. An ideal framework for scene understanding is to use multiple sensors to accumulate their benefits while compensating for their limitations. The next section will present preliminary works exploring senor fusion between LiDAR and RADAR to benefit from both sensor properties.

5.2 Sensor fusion

5.2.1 Introduction

A LiDAR is an active sensor transmitting laser beams in the environment, so it is not affected by lighting conditions as during night. The sensor measures the time delay of the reflected light to be received back, recording the distance and the light, or intensity, of a reflection. Multiple laser beams (usually 16, 32 or 64 depending on the sensor) scan the car's surroundings, generally by considering a 360° FoV. The LiDAR sensor provides a dense 3D point cloud measuring the geometry of a scene with a resolution below the degree for both azimuth and elevation angles. These advantages have brought LiDAR to the forefront in understanding 3D scenes.

The RADAR sensor emits electromagnetic waves, which are not impacted by adverse weather conditions, and records the location, the Doppler and the reflectivity of the objects in the scenes using the received signals (see Sections 2.1 and 2.2). The RADAR point cloud is a lightweight representation that is easy to manipulate in Cartesian coordinates. It is ob-

tained after a pre-processing pipeline degrading the objects' reflections. The RADAR point cloud is a sparse representation composed of about ten points considering a LD RADAR and cannot be used as-is for scene understanding.

The LiDAR sensor is perturbed by adverse weather conditions (fog, rain, snow) because the light is reflected by droplets, creating artifacts [Bijelic *et al.* 2018, Guan *et al.* 2020, Karlsson *et al.* 2021]. Moreover, it only provides a dense point cloud at short range and cannot reach objects as far as RADAR. Consequently, fusing RADAR and LiDAR point clouds should lead to benefit from their respective advantages while compensating for their limitations. As they are both represented in Cartesian coordinates, an early fusion module is easily feasible.

As detailed in Section 3.5 and to the best of our knowledge, there is no related work on RADAR and LiDAR point clouds early fusion to exploit both representations as a single enriched point cloud. In this section, we propose an early fusion module to propagate RADAR information through the LiDAR point cloud by considering the resolution and accuracy of the RADAR sensor as a quantification of its uncertainty.

Section 5.2.2 will describe our propagation and fusion method which will be simulated in Section 5.2.3. Section 5.2.4 will present an application to the nuScenes dataset [Caesar *et al.* 2020]. Finally, Section 5.2.5 will discuss our proposed method and our future work.

5.2.2 Method

The proposed method consists in projecting the RADAR and LiDAR point clouds in a 2D BEV coordinates plane[4]. Then it defines an uncertainty area of each RADAR points w.r.t. the sensor specificity, *i.e.* a zone in which the points are not perfectly positioned. Finally, it propagates the RADAR point properties through the LiDAR points belonging to its uncertainty area and fusing all the points in a unified point cloud.

Let $\mathbf{q}^{\mathrm{R}} \in \mathbb{R}^{m\times 5}$ be a RADAR point cloud of m points, where the i-th point is written $(x_i^{\mathrm{R}}, y_i^{\mathrm{R}}, v_{x,i}, v_{y,i}, \sigma_i^{\mathrm{R}})$, with $(x_i^{\mathrm{R}}, y_i^{\mathrm{R}})$ its Cartesian coordinates, σ_i^{R} the RCS of the reflected signal and $(v_{x,i}, v_{y,i})$ the Doppler vector in Cartesian coordinates, compensated by the ego vehicle velocity.

Let $\mathbf{q}^{\mathrm{L}} \in \mathbb{R}^{n\times 3}$ be a LiDAR point cloud of n points, where the j-th point is written $(x_j^{\mathrm{L}}, y_j^{\mathrm{L}}, \sigma_j^{\mathrm{L}})$, with $(x_j^{\mathrm{L}}, y_j^{\mathrm{L}})$ its Cartesian coordinates and σ_j^{L} the intensity of the reflected light.

The propagation and fusion module creates a single point cloud $\mathbf{q}^{\mathrm{F}} \in \mathbb{R}^{(m+n)\times 6}$, where the k-th point is written $(x_k^{\mathrm{F}}, y_k^{\mathrm{F}}, \sigma_k^{\mathrm{L}}, v_{x,k}, v_{y,k}, \sigma_k^{\mathrm{R}})$ including both $\mathbf{q}^{\mathrm{L}}$ and $\mathbf{q}^{\mathrm{R}}$ information, *i.e.* carrying its 2D Cartesian coordinates, the reflectivity of the LiDAR point, the RCS and the two components of the Doppler vector.

In this section, we introduce the RADAR sensor uncertainty, *i.e.* the quantified uncertainty related to the accuracy and resolution of the sensor measurements. Depending on the RADAR sensor, it will have several modes of transmission and reception of the signals. The modes will define the maximum distance and the FoV that the signal can reach. An

[4]The elevation angle of the LD RADAR is not considered as relevant since the position of the antennas in the sensor does not allow for multiple elevation angles to be recorded.

example of the RADAR specificity is presented in Table 5.5; it corresponds to the sensor used in the nuScenes dataset. This RADAR sensor has three modes: a far-range mode and two short-range modes (either with $\pm 45°$ or $\pm 60°$ FoV). Each one of these modes has its own specificity in resolution, accuracy and maximum measurements in both distance and azimuth angle. Additional details on the application of the presented method are provided in Section 5.2.4. The distance resolution is assumed to be linearly increasing and its accuracy is fixed for each mode. The azimuth resolution and accuracy are fixed for each mode.

By knowing the location of the RADAR point, we can associate a resolution and an accuracy in both distance and azimuth angle. These sensor uncertainty measurements in range and azimuth angle, respectively written δr and $\delta\alpha$, correspond to the minor and major axis of an ellipse in polar coordinates.

Let us consider a RADAR and a LiDAR point cloud. Each point cloud is projected in polar coordinates to propagate the RADAR information in the LiDAR points belonging to each uncertainty ellipse. Let $\mathcal{P}$ be the map defined as

$$\begin{array}{rccc} \mathcal{P}: & \mathbb{R}^{k\times 6} & \longrightarrow & \mathbb{R}^{k\times 6} \\ \mathbf{q}=[\mathbf{q}_0,\mathbf{q}_1,\mathbf{q}_2,\ldots,\mathbf{q}_5] & \longmapsto & \tilde{\mathbf{q}}=\left[\tan^{-1}(\mathbf{q}_1/\mathbf{q}_0),\sqrt{\mathbf{q}_0^2+\mathbf{q}_1^2},\mathbf{q}_2,\ldots,\mathbf{q}_5\right] \end{array} \tag{5.5}$$

transforming an arbitrary point cloud $\mathbf{q}$ of k points in Cartesian coordinates into a point cloud $\tilde{\mathbf{q}}$ in polar coordinates w.r.t. the sensor coordinates. We will note $(\mathbf{x}^{\mathrm{R}},\mathbf{y}^{\mathrm{R}})$ the 2D Cartesian coordinates of the RADAR point cloud $\mathbf{q}^{\mathrm{R}}$, and $(\boldsymbol{\alpha}^{\mathrm{R}},\mathbf{r}^{\mathrm{R}})$ their corresponding 2D polar coordinates.

In practice, we initialise the LiDAR and RADAR point cloud with constant values for their respective missing features to use $\mathcal{P}$. *E.g* the RADAR point cloud will have an additional dimension with constant values noted σ^{cst} filling the missing information of a LiDAR point. The LiDAR point cloud will have three additional dimensions written $\sigma^{\mathrm{cst}},(v_x^{\mathrm{cst}},v_y^{\mathrm{cst}})$ filling the missing information of a RADAR point cloud, reflectively the RCS and the Doppler components. This way, the two points clouds $\mathbf{q}^{\mathrm{L}}\in\mathbb{R}^{n\times 6}$ and $\mathbf{q}^{\mathrm{R}}\in\mathbb{R}^{m\times 6}$ are projected in polar coordinates, respectively written $\mathcal{P}(\mathbf{q}^{\mathrm{L}})=\tilde{\mathbf{q}}^{\mathrm{L}}$ and $\mathcal{P}(\mathbf{q}^{\mathrm{R}})=\tilde{\mathbf{q}}^{\mathrm{R}}$.

Considering the i-th RADAR point $(\alpha_i^{\mathrm{R}},r_i^{\mathrm{R}})$ in polar coordinates, its uncertainty ellipse is defined as $\mathcal{B}_{(\delta\alpha,\delta r)}(\alpha_i^{\mathrm{R}},r_i^{\mathrm{R}})$, a ball centered on $(\alpha_i^{\mathrm{R}},r_i^{\mathrm{R}})$, where $\delta\alpha,\delta r$ are respectively its azimuth major axis and its range minor axis. On the one hand, the RADAR properties of the point $(\alpha_i^{\mathrm{R}},r_i^{\mathrm{R}})$ are propagated to the LiDAR points belonging to this ellipse. On the other hand, the properties of the LiDAR points belonging to the ellipse will be averaged to be propagated to the RADAR point. In other words, we propose to propagate the RCS and the Doppler components, respectively noted σ_i^{R} and $(v_{x,i},v_{y,i})$, corresponding to the RADAR point $(\alpha_i^{\mathrm{R}},r_i^{\mathrm{R}})$, to the LiDAR points in its ellipse. The reflected intensity vector of the LiDAR points in the ellipse, noted $\boldsymbol{\sigma}^{\mathrm{L}}$, is averaged and associated to the RADAR point at the center of the ellipse. Note that the proposed method is asymmetric to adapt the difference in sparsity between the RADAR and LiDAR point clouds.

Considering the j-th LiDAR point $(\alpha_j^{\mathrm{L}},r_j^{\mathrm{L}})$ in polar coordinates, it belongs to the

RADAR uncertainty ellipse $\mathcal{B}_{(\delta\alpha,\delta r)}(\alpha_i^{\mathrm{R}}, r_i^{\mathrm{R}})$ if

$$\frac{(\alpha_j^{\mathrm{L}} - \alpha_i^{\mathrm{R}})^2}{\delta\alpha} + \frac{(r_j^{\mathrm{L}} - r_i^{\mathrm{R}})^2}{\delta r} \leq 1 \tag{5.6}$$

is verified. The LiDAR points belonging to the ellipse are grouped in a sub point cloud written $\tilde{\mathbf{q}}_+^{\mathrm{L}}$ and removed from the initial LiDAR point cloud $\tilde{\mathbf{q}}^{\mathrm{L}}$.

After the propagation step, three distinct point clouds are obtained: $\tilde{\mathbf{q}}_+^{\mathrm{F}}$ the sub-LiDAR point cloud with propagated RADAR information, $\tilde{\mathbf{q}}^{\mathrm{F}}$ the sub-LiDAR point cloud containing the remaining points and $\tilde{\mathbf{q}}^{\mathrm{R}}$ the RADAR point cloud. A single point cloud $\tilde{\mathbf{q}}^{\mathrm{F}} = [(\tilde{\mathbf{q}}_+^{\mathrm{L}})^\top, (\tilde{\mathbf{q}}^{\mathrm{L}})^\top, (\tilde{\mathbf{q}}^{\mathrm{R}})^\top]^\top \in \mathbb{R}^{(m+n)\times 6}$ is created by stacking them, containing both the RADAR and LiDAR point clouds with the propagated information or constant values where appropriate. The final point cloud is projected back in Cartesian coordinates: $\mathbf{q}^{\mathrm{F}} = \mathcal{P}^{-1}(\tilde{\mathbf{q}}^{\mathrm{F}})$. The entire method is detailed in Algorithm 2. A simulation of this method will be presented in the following section.

Algorithm 2 Proposed propagation and fusion module. We note $\mathbf{A}_{i,j}$ the element of the matrix $\mathbf{A}$ at the i-th row and j-th column, $\mathbf{A}_{.,j}$ the elements of the matrix $\mathbf{A}$ corresponding to the all rows of the j-th column, and $\mathbf{A}_{i,.}$ the elements of the matrix $\mathbf{A}$ corresponding to all the columns of the i-th row.

Require: $\mathbf{q}^{\mathrm{R}} \in \mathbb{R}^{m \times 5}, \mathbf{q}^{\mathrm{L}} \in \mathbb{R}^{n \times 3}$

m the number of RADAR points, n the number of LiDAR points.

$[\boldsymbol{\sigma}^{\mathrm{cst}}, \mathbf{v}_x^{\mathrm{cst}}, \mathbf{v}_y^{\mathrm{cst}}] \leftarrow [\mathbf{c}_0, \mathbf{c}_1, \mathbf{c}_2]$ # Set constant values.

$\mathbf{q}^{\mathrm{R}} \leftarrow [\mathbf{q}^{\mathrm{R}}_{.,0}, \mathbf{q}^{\mathrm{R}}_{.,1}, \boldsymbol{\sigma}^{\mathrm{cst}}, \mathbf{q}^{\mathrm{R}}_{.,2}, \mathbf{q}^{\mathrm{R}}_{.,3}, \mathbf{q}^{\mathrm{R}}_{.,4}]$ # Set $\mathbf{q}^{\mathrm{R}} \in \mathbb{R}^{m \times 6}$ with cst: $\boldsymbol{\sigma}^{\mathrm{cst}}$.

$\mathbf{q}^{\mathrm{L}} \leftarrow [\mathbf{q}^{\mathrm{L}}_{.,0}, \mathbf{q}^{\mathrm{L}}_{.,1}, \mathbf{q}^{\mathrm{L}}_{.,2}, \mathbf{v}_x^{\mathrm{cst}}, \mathbf{v}_y^{\mathrm{cst}}, \boldsymbol{\sigma}^{\mathrm{cst}}]$ # Set $\mathbf{q}^{\mathrm{L}} \in \mathbb{R}^{n \times 6}$ with cst: $\boldsymbol{\sigma}^{\mathrm{cst}}$, $\mathbf{v}_x^{\mathrm{cst}}$, $\mathbf{v}_y^{\mathrm{cst}}$.

$\tilde{\mathbf{q}}^{\mathrm{R}}, \tilde{\mathbf{q}}^{\mathrm{L}} \leftarrow \mathcal{P}(\mathbf{q}^{\mathrm{R}}), \mathcal{P}(\mathbf{q}^{\mathrm{L}})$ # Project $\tilde{\mathbf{q}}^{\mathrm{R}}$ and $\tilde{\mathbf{q}}^{\mathrm{L}}$ in polar coordinates.

$h \leftarrow 0$ # Set counter to 0.

for $i = 0$ **to** $m - 1$ **do** # m RADAR points.

  $[\alpha_i^{\mathrm{R}}, r_i^{\mathrm{R}}, \sigma_i^{\mathrm{cst}}, v_{x,i}, v_{y,i}, \sigma_i^{\mathrm{R}}] \leftarrow \tilde{\mathbf{q}}^{\mathrm{R}}_{i,.}$ # Set the RADAR scalar values.

  $(\delta\alpha, \delta r) \leftarrow$ get_uncertainty$(\alpha_i^{\mathrm{R}}, r_i^{\mathrm{R}})$ # Get the ellipse parameters.

  $\mathbf{A}$: empty matrix with 6 columns # Buffer of enriched LiDAR points.

  $k \leftarrow 0$ # Set counter to 0.

  for $j = 0$ **to** $|\tilde{\mathbf{q}}^{\mathrm{L}}| - 1$ **do** # Number of LiDAR points in $\tilde{\mathbf{q}}^{\mathrm{L}}$.

    $[\alpha_j^{\mathrm{L}}, r_j^{\mathrm{L}}, \sigma_j^{\mathrm{L}}, v_{x,i}^{\mathrm{cst}}, v_{y,i}^{\mathrm{cst}}, \sigma^{\mathrm{cst}}] \leftarrow \tilde{\mathbf{q}}^{\mathrm{L}}_{j,.}$ # Set the LiDAR scalar values.

    if $\frac{(\alpha_j^{\mathrm{L}} - \alpha_i^{\mathrm{R}})^2}{\delta\alpha} + \frac{(r_j^{\mathrm{L}} - r_i^{\mathrm{R}})^2}{\delta r} \leq 1$ **then** # Check if the LiDAR point is in the ellipse

      $\mathbf{A}_{k,.} \leftarrow [\alpha_j^{\mathrm{L}}, r_j^{\mathrm{L}}, \sigma_j^{\mathrm{L}}, v_{x,i}, v_{y,i}, \sigma_i^{\mathrm{R}}]$ # Add the point to the enriched buffer.

      $k \leftarrow k + 1$ # Update the counter.

  if $k \neq 0$ **then** # Check if LiDAR points have been added to the buffer.

    $\overline{\sigma} = \mathrm{mean}(\mathbf{A}_{.,2})$ # Compute the mean of LiDAR's σ in the buffer.

    $\tilde{\mathbf{q}}^{\mathrm{R}}_{i,.} \leftarrow [\alpha_i^{\mathrm{R}}, r_i^{\mathrm{R}}, \overline{\sigma}, v_{x,i}, v_{y,i}, \sigma_i^{\mathrm{R}}]$

Update the RADAR point with the averaged LiDAR's σ.

  $\tilde{\mathbf{q}}^{\mathrm{L}}_{+,h:h+k,.} \leftarrow \mathbf{A}$ # Add the buffer to the enriched LiDAR point cloud.

  $\tilde{\mathbf{q}}^{\mathrm{L}} \leftarrow \tilde{\mathbf{q}}^{\mathrm{L}}$.delete$(\mathbf{A})$ # Remove the buffer from initial LiDAR point cloud.

  $h \leftarrow h + k$ # Update the counter.

$\tilde{\mathbf{q}}^{\mathrm{F}} \leftarrow \begin{bmatrix} \tilde{\mathbf{q}}^{\mathrm{R}} \\ \tilde{\mathbf{q}}^{\mathrm{L}}_{+} \\ \tilde{\mathbf{q}}^{\mathrm{L}} \end{bmatrix}$ # Fuse the point clouds.

$\mathbf{q}^{\mathrm{F}} \leftarrow \mathcal{P}^{-1}(\tilde{\mathbf{q}}^{\mathrm{F}})$ # Map the fused point cloud in Cartesian coordinates.

5.2.3 Simulation

The proposed method detailed in Algorithm 2 has been simulated with randomly generated point clouds. Let $n = 2000$ and $m = 20$ be the number of points in the LiDAR and RADAR point clouds respectively. For both point clouds, random Cartesian coordinates are drawn as $\mathbf{x} \sim \mathcal{U}([-50, 50])$ and $\mathbf{y} \sim \mathcal{U}([0, 250])$, the RADAR sensor position has been fixed to $(0, 0)$. The order of magnitude of the size of each point cloud has been chosen according to real data recorded from a 32-beam LiDAR and LD RADAR.

The simulation considers the specificity of the RADAR sensor used in the nuScenes dataset [Caesar *et al.* 2020] described in Table 5.5. Figure 5.9(a) illustrates the randomly drawn LiDAR and RADAR point clouds, respectively in yellow and green. The Out-of-Scope (OS) RADAR points are not belonging to any RADAR mode considering the geometric priors of the sensor specificity; they are excluded from the simulation. Both point clouds are then mapped in polar coordinates and the uncertainty ellipse of each RADAR point is computed in polar coordinates as depicted in Figure 5.9(b)[5]. The resolution and accuracy in distance and azimuth angle are defined according the method presented in Section 5.2.2 and with the sensor specificity described in Table 5.5. As detailed in Algorithm 2, the LiDAR points belonging to an ellipse obtain the propagated information of the RADAR point at the center of the ellipse. These points are denoted as "Propag. RADAR" in Figure 5.9(c). Finally, Figure 5.9(d) illustrates in green both the RADAR points and the LiDAR points which have benefited from the propagation method. The aim of the fusion is to consider both green and yellow points as a single point cloud. The simulation shows that the proposed method succeeds to locally propagate the RADAR information in a denser point cloud by only considering the resolution and the accuracy of the sensor. The following section will present the application of this method to a real dataset.

5.2.4 Application to the nuScenes dataset

The nuScenes dataset [Caesar *et al.* 2020] is considered as one of the largest automotive dataset publicly available containing radar data. It has been briefly described in Section 3.2 and Table 3.1. It is composed of 5.5 hours of recorded sequences in two countries including night and rain weather conditions. The car is mounted with a 32-beam LiDAR, 6 cameras and 5 LD ARS 408-21 RADARs[6] recording simultaneously with a 360° FoV. The parameters and settings of each RADAR sensor are described in Table 5.5. The authors proposed 3D bounding boxes annotations with tracking metrics and a recent extension containing semantic segmentation annotations while considering 23 classes and 8 attributes. The dataset is composed of 1000 scenes divided in three sub-sets: "mini", "trainval" and "test". In our preliminary work, only the "mini" dataset containing 10 scenes has been explored yet.

Each sensor used to record the data has its own orientation axis as described in Figure 5.5 of Appendix B.2.1. Therefore our method has been applied iteratively on each RADAR orientation axis by projecting the LiDAR point cloud in the same coordinates system. By

[5] Note that a scaling factor of 2.0 has been applied to the ellipses for visualization purpose. The qualitative results presented in the next section are obtained with a scaling factor fixed to 1.0.

[6] https://www.continental-automotive.com/

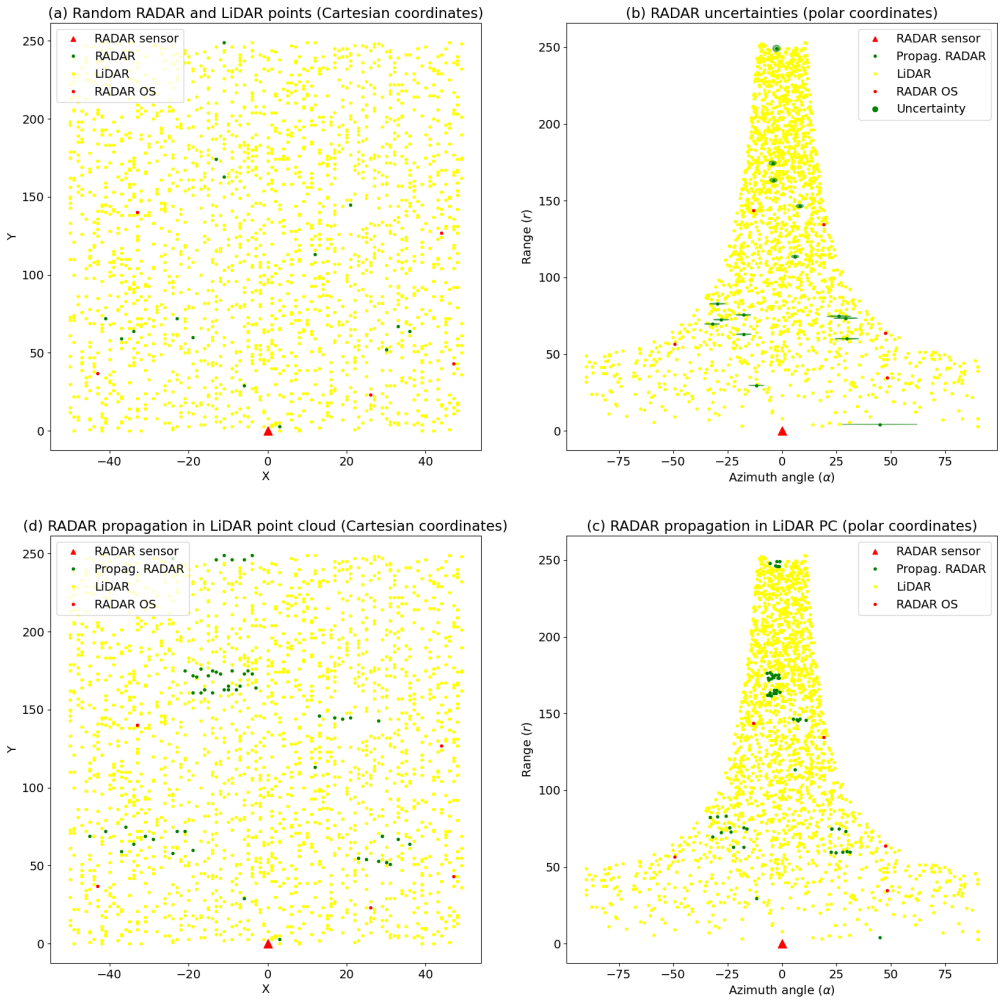


Figure 5.9: **Simulation of the LiDAR and RADAR point cloud propagation and fusion method**. (a) The LiDAR and RADAR point clouds are randomly drawn in Cartesian coordinates, the RADAR points are filtered according to the sensor specificity (Out-of-Scope (OS) points). (b) The point clouds are transformed in polar coordinates with the RADAR uncertainty ellipses illustrated with a scaling factor of 2.0. (c) The RADAR point information is propagated to the LiDAR points and (d) transformed back in Cartesian coordinates. Note that the "Propag. RADAR" point cloud groups the RADAR points and LiDAR points in the uncertainty areas.

doing so, the considered RADAR sensor is at the origin of its coordinates system and the polar coordinates of each point cloud will be computed w.r.t. its position. The procedure described in Section 5.2.2 and Algorithm 2 is applied independently for each RADAR sensor while considering the same 360° LiDAR point cloud.

Figure 5.10 (left) illustrates a scene with a LiDAR point cloud in orange and the RADAR point cloud in green. Each RADAR point is characterized by a black arrow corre-

Specificity	RADAR mode		
	Far range	Near range (45°)	Near range (60°)
Azimuth angle FoV (deg)	±9°	±45°	±60°
Azimuth angle resolution (deg)	1.6°	4.5°	12.3°
Accuracy azimuth angle (deg)	±0.1°	±1.0°	±5.0°
Distance range (m)	[0.2, 250]	[0.2, 100]	[0.2, 20]
Distance range resolution (m)	[0, 1.79]	[0, 0.39]	[0, 0.39]
Accuracy distance (m)	±0.40	±0.10	±0.10

Table 5.5: **Parameters and settings of the RADAR sensors used in the nuScenes dataset**.

sponding to its Doppler compensated by the ego-vehicle velocity. The light-blue boxes and their corresponding arrows are the annotated objects in the scenes with their velocity. By applying our method, we propagate the RADAR information to the LiDAR points according to the sensor uncertainty defined by its specificity (Table 5.5). The propagated points combined with the RADAR point clouds are depicted in green in Figure 5.10 (middle). By propagating the RADAR information in the LiDAR point cloud, we show qualitatively in Figure 5.10 (right) that our method helps to obtain a denser Doppler information in the combined point cloud. Additional qualitative results are presented in Figure B.10 of Appendix B.2.2 leading to the same conclusions. Knowing that these points also carry the RCS and the LiDAR reflection intensity, we hope that our proposed propagation and fusion module will help to improve detection and classification of objects considering point cloud representations. The following section will discuss our results and future work.

5.2.5 Discussions and future work

In this section, we presented our propagation and fusion module quantifying the measurement uncertainty of the RADAR sensor. By considering both RADAR and LiDAR point clouds, we propose to propagate the RADAR point properties to the LiDAR points belonging to its uncertainty area. The other way around, the RADAR point will also carry the average reflection of all the LiDAR points in its uncertainty area. Our module proposes to fuse both point clouds to benefit from the density of the LiDAR data and the Doppler information of the RADAR data while containing the reflectivity information of both sensors.

However, we notice that our method is impacted by ghost and multi-path RADAR reflections. In these cases, the propagation may assign a Doppler or RCS to an object which it does not belong to. Inspired by the work of [Kopp *et al.* 2021], rule-based method could be integrated to filter the RADAR point cloud before the propagation step. Another limitation of our method is to consider that each LiDAR point has an equivalent probability to belong to the uncertainty ellipse of a RADAR point. We will tackle this issue by considering this ellipse has a multi-variate Gaussian probability distribution, centered on the RADAR point and with a variance-covariance matrix defined by its distance and azimuth angle axis (previously noted δr and $\delta\alpha$). This way, each LiDAR point will be associated to a probability to belong to this ellipse.

The aim of this work is to benefit from both LiDAR and RADAR point clouds to improve scene understanding. In our future work, we will compare a point cloud based

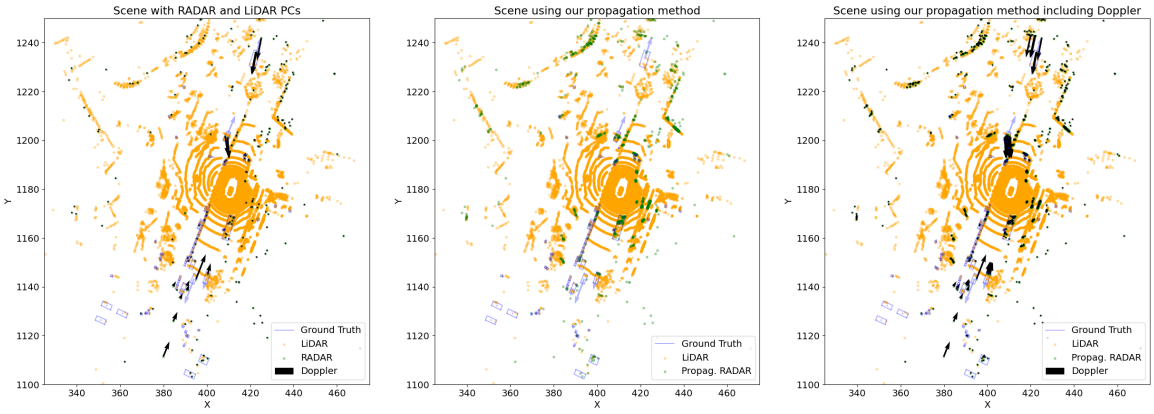


Figure 5.10: **Qualitative results on the nuScenes dataset of our propose propagation and fusion module**. (Left) Scene in Bird's Eye View representation with LiDAR and RADAR point clouds. (Middle) The point cloud illustrated in green groups the RADAR and propagated RADAR points with Doppler and reflectivities. (Right) The propagated Doppler information is illustrated with black arrows to distinguish moving objects.

deep learning algorithm (*e.g.* PointNet [Qi *et al.* 2017a] or PointNet++ [Qi *et al.* 2017b]) while being trained with either a single sensor point cloud or with our enhanced point cloud. In a second approach, we will explore self-supervised learning by training a model to predict a modality of our enhanced point cloud and fine-tune it on a downstream task. *E.g.* a model trying to predict the Doppler information will learn to discriminate moving and static objects while using a single timestamp point cloud. The neural network will be then fine-tuned to improve an object detection or a segmentation task.

5.3 Conclusions

This chapter has described innovative approaches for RADAR scene understanding using deep learning algorithms. In Section 5.1, we proposed a method for multi-view RADAR semantic segmentation based on the exploitation of the CARRADA dataset presented in Section 4.3. We have detailed several deep learning architectures, with their associated loss functions, outperforming competing methods with significantly fewer parameters. Our method is the only capable to simultaneously perform RD and RA semantic segmentation. We have introduced an unsupervised Coherence loss to enforce the spatial coherence between RADAR views during training. Our proposed method has achieved the state of the art performances on the CARRADA dataset and on the recently published RADDet dataset. Conducted experiments have shown that our method is able to segment objects well in multi-views of the RADAR tensor in both simple and complex urban scenes. This second main contribution of the thesis was presented at the International Conference of Computer Vision (ICCV). In our future work, we will improve the aggregation of the RAD tensor to avoid object signature attenuation. We will also explore an extended Coherence loss formulation to introduce the ground truth and better quantify the spatial errors.

In Section 5.2, we presented a preliminary work on LiDAR and RADAR point cloud fusion. Our fusion and propagation method quantifies the uncertainty of the polar coordinate point location by considering the specificity of the RADAR sensor. On one hand, our method propagates the Doppler and RCS information of a RADAR point to the LiDAR points contained in its uncertainty area. On the other hand, it propagates the average LiDAR reflections contained in this area to the RADAR point. This way, an enriched point cloud is proposed, containing the velocity and reflection information of the RADAR point cloud while benefiting from the density of the LiDAR point cloud. In our future work, we will train deep neural network architectures specialized in point cloud representation with the proposed enriched point cloud to improve scene understanding tasks, *e.g.* object detection and semantic segmentation. In addition, we will explore self-supervised learning schemes applied to multi-sensor fusion. In particular, we will exploit the enriched point cloud to learn a neural network for Doppler prediction. We hope that this method will learn representations to distinguish moving objects with any additional annotations. This model will be then fine-tuned on well-known scene understanding tasks to show quantitative improvement from this process.

Further details on our future work will be provided in Section 7.2. In the following chapter, we will present a collaborative project proposing RADIal, a recent dataset with synchronized camera, LiDAR and raw HD RADAR data with annotations for 2D object detection and free driving space segmentation. We will also present a deep learning architecture for multi-task learning while estimating a part of the costly RADAR pre-processing.

CHAPTER 6

High-definition RADAR

Contents

6.1 Motivations 112
6.2 RADIal dataset 113
6.3 Proposed method 115
6.3.1 MIMO pre-encoder 115
6.3.2 FPN encoder 117
6.3.3 Range-Angle decoder 117
6.3.4 Multi-task learning 117
6.4 Experiments and Results 119
6.4.1 Training details 119
6.4.2 Baselines 119
6.4.3 Evaluation metric 120
6.4.4 Performance analysis 120
6.4.5 Complexity analysis 121
6.5 Conclusions and discussions 122

HD RADAR reaches a high angular resolution thanks to the large number of virtual antennas that it is composed of. However, it generates a large volume of data, directly impacting real-time applications. This chapter details RADIal, a unique dataset with raw HD RADAR data synchronized with LiDAR and camera. Annotations for 2D object detection and free driving space segmentation, *i.e.* classify each pixel of a representation as free to be driven or not, are generated with a semi-automatic pipeline. It also presents, FFT-RadNet, a multi-task deep neural network architecture processing raw RADAR signals and succeeding to estimate the RADAR processing steps avoiding large computational cost.

This chapter is organized as follows: Section 6.2 introduces the RADIal dataset, Section 6.3 details the proposed method, Section 6.4 presents the experimental results, finally Section 6.5 concludes.

This chapter presents a work carried out in collaboration with Julien Rebut and mainly inspired from our article published at the Conference on Computer Vision and Pattern Recognition (CVPR) [Rebut *et al.* 2022].

6.1 Motivations

Recent progress towards HD Imaging RADAR has driven the angular resolution below the degree, thus approaching LiDAR performance. By using dense virtual antenna arrays, these sensors achieve high angular resolution both in azimuth and elevation (horizontal and vertical angular positions, respectively) and produce denser RADAR point clouds. The previous chapters focused on LD RADARs with a single elevation pitch due to the geometry of their antennas. As a consequence, the previously noted RA view represented the Azimuth angle. In this chapter, we distinguish the Elevation angle deduced from adjacent pairs of antennas in the vertical axis so the RA representation includes both Azimuth and Elevation angles. As a consequence, HD RADAR representations are cumbersome, with an order of magnitude larger than the data recorded by a LD RADAR.

As seen in Chapter 3, most of the recent works exploit the Range-Azimuth representation of the RADAR data (either in polar or cartesian coordinates). Similar to a BEV (see Figure 6.1), this representation is easy to interpret and allows simple data augmentation with translations and rotations. However, one barely-mentioned drawback is that the generation of the RA RADAR view incurs significant processing costs (tens of GOPS, see Section 6.4.5), which compromises its viability on embedded hardware. While novel HD RADARs offer better resolution, they make this computational complexity issue even more acute.

As detailed in Section 2.2, the AoA is deduced by applying an inverse FFT on the pairs of antennas axis of the recorded tensor in the frequency domain (see Figure 2.2). An alternative is to correlate the RD in the complex domain with a calibration matrix to estimate both the azimuth and the elevation angles. The complexity of this operation for a single point of the RD tensor is $\mathcal{O}(N_{\text{Tx}} N_{\text{Rx}} B_{\text{A}} B_{\text{E}})$, where N_{Tx} is the number of transmitting antennas, N_{Rx} the number of receiving antennas, B_{A} and B_{E} are respectively the number of discretization bins for azimuth and elevation angles in the calibration matrix. For a 4D representation in Range-Azimuth-Elevation-Doppler (RAED), this operation would need to be performed for each point of the RD tensor[1]. Considering an embedded HD RADAR, traditional signal processing cannot be applied as it is too resource greedy in terms of both computation requirements and memory footprint. For driving assistance systems, there is therefore the challenge of increasing radar's angular accuracy while keeping the processing costs under control.

In this chapter, we propose a unique dataset, nick-named RADIal[2] for "RADAR, LiDAR *et al.*", with the first raw HD RADAR dataset including several other automotive-grade sensors, as described in Table 3.1. It is composed of 2 hours of raw data from synchronized automotive-grade sensors (camera, LiDAR, HD radar), annotated for object detection and free space segmentation, and collected in various environments (city street, highway, countryside road). We also propose a novel HD RADAR sensing model, FFT-RadNet, that eliminates the overhead of computing the RAD 3D tensor, learning instead to recover angles from an RD view. FFT-RadNet is trained both to detect vehicles and to

[1] Considering a HD RADAR with 0.2° of azimuth resolution over 180° of horizontal FoV and 11 elevations, it would require 498 Giga FLoating-point Operations Per Second (GFLOPS) to be computed.

[2] RADIal is available online at `https://github.com/valeoai/RADIal`.

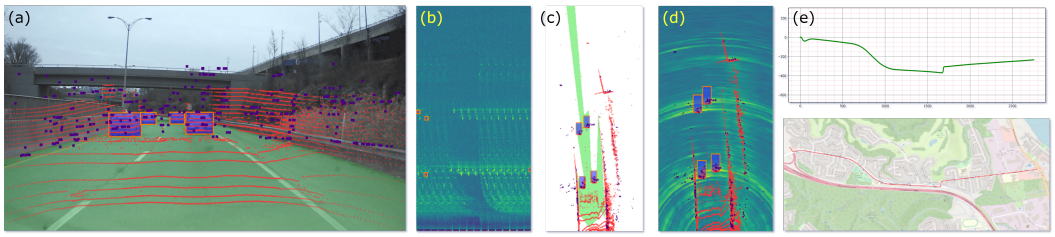


Figure 6.1: **Overview of our RADIal dataset**. RADIal includes a set of 3 sensors (camera, LiDAR, high-definition radar) and comes with GPS and vehicle's CAN traces; 25,000 synchronized samples are recorded in raw format. (a) Camera image with projected LiDAR point cloud in red and RADAR point cloud in indigo, vehicle annotation in orange and free driving space annotation in green; (b) RADAR power spectrum with bounding box annotations; (c) Free driving space annotation in bird-eye view, with annotated vehicle bounding boxes in orange, RADAR point cloud in indigo and LiDAR point cloud in red; (d) Range-Azimuth map in Cartesian coordinates overlayed with RADAR point cloud and LiDAR point cloud; (e) GPS trace in red and odometry trajectory reconstruction in green.

segment free driving space. On both tasks, it competes with recent scene understanding models while requiring less computations and memory.

6.2 RADIal dataset

As depicted in Table 3.1, publicly-available datasets do not provide raw RADAR signal, either for LD RADAR nor for HD RADAR. Therefore, we built RADIal, a new dataset to allow research on automotive HD RADAR.

As RADIal includes 3 sensor modalities –camera, RADAR and LiDAR–, it should also permit one to investigate the fusion of HD RADAR with other common sensors. The specifications of the sensor suite are detailed in Table 6.2. Except for the camera, all sensors are automotive-grade qualified. On top of that, the Global Positioning System (GPS) position and full Controller Area Network (CAN bus) of the vehicle (including odometry) are also provided. Sensor signals were recorded simultaneously in a raw format, without any signal pre-processing. In the case of the HD RADAR, the raw signal is the ADC. From this ADC data, all conventional RADAR representations can be generated: RAD tensor, RA and RD views or point cloud.

Central to the proposed RADIal dataset, our HD RADAR is composed of $N_{\text{Rx}} = 16$ receiving antennas and $N_{\text{Tx}} = 12$ transmitting antennas, leading to $N_{\text{Rx}} \cdot N_{\text{Tx}} = 192$ virtual antennas. This virtual-antenna array enables reaching a high azimuth angular resolution while estimating objects' elevation angles as well. As the RADAR signal is difficult to interpret by annotators and practitioners alike, a 16-layer automotive-grade LiDAR and a 5 Mpix RGB camera are also provided. The camera is placed below the interior mirror behind the windshield while the RADAR and the LiDAR are installed in the middle of the front ventilation grid, one above the other. The three sensors have parallel horizontal lines of sight, pointing in the driving direction. Their extrinsic parameters are provided together

		HD Radar	LiDAR	Camera
FOV	Range	103 m	150 m	–
	Azimuth	180°	133°	100°
	Elevation	12°	10°	75°
resolution	Range	0.2 m	0.1 m	
	Azimuth	0.1°	0.125-0.25°	2592 px
	Elevation	1°	0.6°	1944 px
	Velocity	$0.1\ \mathrm{m{\cdot}s^{-1}}$	–	–
Frame rate		5fps	25fps	30fps
Height above ground		80 cm	42 cm	145 cm

Table 6.1: **Specifications of the RADIal's sensor suite**. The main characteristics of the HD RADAR, the LiDAR and the camera are reported. Their synchronized signals are complemented by GPS and CAN bus information.

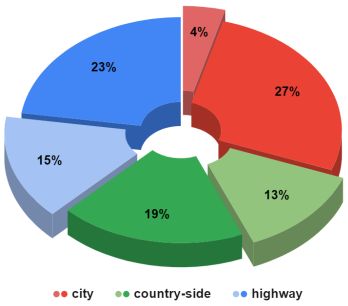


Table 6.2: **Scene-type proportions in RADIal**. The dataset contains 91 sequences in total, captured on city streets, highway or country-side roads, for a total of 25k synchronized frames (dark colors), out of which 8,252 are labelled (light colors).

with the dataset. RADIal also offers synchronized GPS and CAN bus traces which give access to the geo-referenced position of the vehicle as well as its driving information such as speed, steering wheel angle and yaw rate. The sensors' specifications are detailed in Table 6.2.

RADIal contains 91 sequences of about 1-4 minutes, for a total of 2 hours. This amounts to approximately 25,000 synchronized frames in total, out of which 8,252 are annotated with 9,550 vehicles. These sequences are categorized in highway, country-side and city driving. The distribution of the sequences is indicated in Figure 6.2.

The annotation of the RADAR signal is hard to achieve as the RD representation is not meaningful for the human eye. Vehicle detection labels were first generated automatically using supervision from the camera and LiDAR. A RetinaNet model [Lin *et al.* 2017b] was used to extract object proposals from the camera. Then, these proposals were validated when both RADAR and LiDAR agree on the object position from their respective point cloud. Finally, manual verification was conducted to reject or validate the labels. The free-space annotation was done fully automatically on the camera images. A DeepLabV3+ [Chen *et al.* 2018b], pre-trained on Cityscape, has been fine-tuned with 2 classes (*free* or *occupied*) on a small manually-annotated part of our dataset.

This model segmented each video frame and the obtained segmentation mask was projected from the camera's coordinate system to the radar's one thanks to known calibration. Finally, already available vehicle bounding boxes were subtracted from the free-space mask. The quality of the segmentation mask is limited due to the automatic method we employed and to the projection inaccuracy from camera to real world. In the next section, the proposed FFT-RadNet architecture and its associated multi-task loss will be presented.

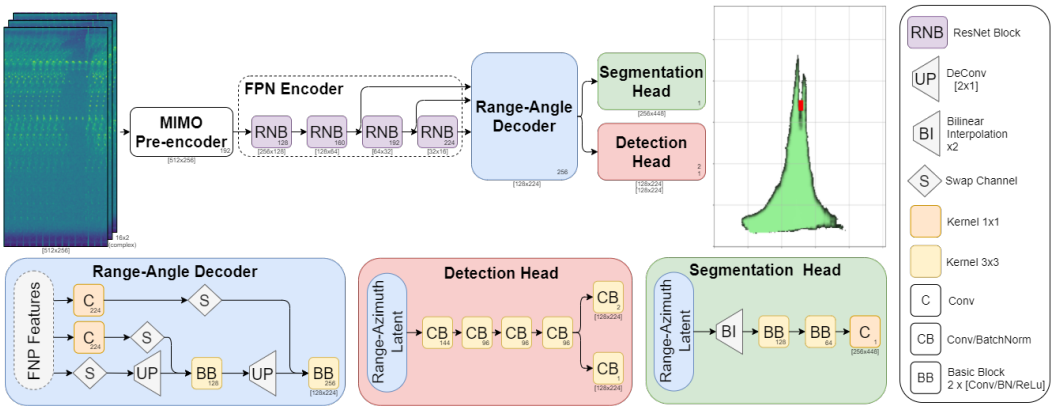


Figure 6.2: **Overview of FFT-RadNet**. FFT-RadNet is a lightweight multi-task architecture. It does not use any Range-Angle maps or Range-Angle-Doppler tensor which would require costly pre-processing. Instead, it leverages complex Range-Doppler containing all the range, azimuth and elevation information. This data is de-interleaved and compressed by the MIMO pre-encoder. A Feature Pyramid Network encoder extracts a pyramid of features which the Range-Angle decoder converts into a latent Range-Azimuth representation. Based on this representation, multi-task heads finally detect vehicles (red rectangle) and predict the free driving space (green shape) in a Bird's Eye View map illustrated in the image on the right.

6.3 Proposed method

Our approach has been motivated by automotive constraints: automotive-grade sensors must be used and only limited processing/memory resources are available on the embedded hardware. In this context, the RD is the only representation that is practical for HD RADAR. Based on it, we propose a multi-task architecture, compatible with above requirements, which is composed of five blocks (see Figure 6.2):

- A pre-encoder reorganizing and compressing the RD tensor into a meaningful and compact representation;
- A shared FPN encoder combining low-resolution semantic information with high-resolution details;
- A RA decoder building a Range-Azimuth latent representation from the feature pyramid;
- A detection head localizing vehicles in Range-Azimuth coordinates;
- A segmentation head predicting the free driving space.

6.3.1 MIMO pre-encoder

As explained in Section 2.1, the MIMO configuration implies one complex RD representation per receiver after the Range-FFT and the Doppler-FFT. This results in a complex

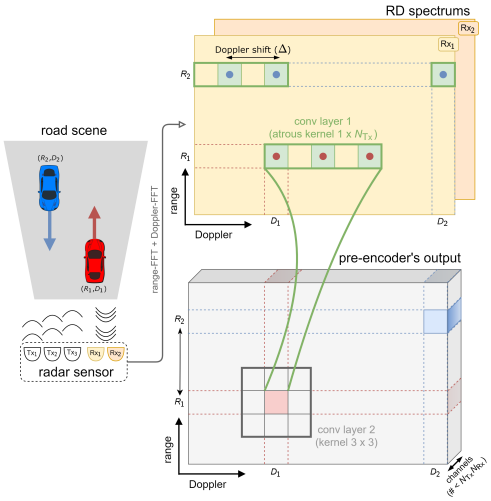


Figure 6.3: **Trainable MIMO pre-encoder**. Considering three transmitters (N_{Tx}=3) and two receivers (N_{Rx}=2), an object's signature is visible N_{Tx} times in the Range-Doppler representation. The pre-encoder uses atrous convolutions to organize and compress signatures in fewer than $N_{\text{Tx}} \cdot N_{\text{Rx}}$ output channels.

3D tensor of dimension $(B_{\text{R}}, B_{\text{D}}, N_{\text{Rx}})$, where B_{R} and B_{D} are the number of discretization bins for range and Doppler respectively. It is important to understand how a given reflecting object, say a car in front, appears in this data. Denote R the actual radial distance of this object to the RADAR and D its relative radial velocity expressed in Doppler effect. For each receiver, its signature will be visible N_{Tx} times, one per transmitter. More specifically, it will be measured at RD positions $(R, (D + k\Delta)[D_{\text{max}}])_{k=1\cdots N_{\text{Tx}}}$, where Δ is the Doppler shift (induced by the phase shift $\Delta_\phi = \phi(t)$ in the transmitted signal, see Section 2.1) and D_{max} is the largest Doppler that can be measured. The measured Doppler values are modulo this maximum. This phenomena is illustrated in Figure 6.1(b).

This signal intricacy calls for a rearrangement of the RD tensor that will facilitate a subsequent exploitation of the MIMO information (to recover angles) while keeping data volume under control. To this end, we propose a new trainable pre-encoder illustrated in Figure 6.3 that performs a compact reorganization of the input tensor. In order to handle its specific structure along the Doppler axis, we use first a suitably-defined atrous convolution layer that gathers Tx and Rx information at the right positions. The size of its kernel for one input channel is $1\times N_{\text{Tx}}$, hence defined by the number of Tx antennas, and its dilation amounts to $\delta = \frac{\Delta B_{\text{D}}}{D_{\text{max}}}$, the number of Doppler bins corresponding to the Doppler shift Δ. The number of input channels is the number N_{Rx} of Rx antennas. A second convolution layer, with a 3×3 kernel, learns how to combine these channels and compresses the signal. The two-layer pre-encoder is trained end-to-end with the rest of the proposed architecture.

6.3.2 FPN encoder

Using a pyramidal structure to learn multi-scale features is a common practice in object detection and semantic segmentation as detailed in Sections 2.6.2 and 2.6.3. Our FPN architecture uses 4 blocks composed of 3, 6, 6, 3 residual layers [He *et al.* 2016] respectively. The feature maps of these residual blocks form the feature pyramid. This encoder has been optimized considering the nature of the data while controlling its complexity. The channel dimensions are chosen to encode at best the azimuth angle over the entire distance range (*i.e.*, high resolution and narrow field of view at far range, low resolution and wider field of view at near range).

To prevent losing the signature of small objects (typically few pixels in the RD representation), the FPN encoder performs a 2×2 down-sampling per block, leading to a total reduction of the tensor size by a factor of 16 in height and width. For similar reasons and to avoid overlaps between adjacent Tx's, it uses 3×3 convolution kernels.

6.3.3 Range-Angle decoder

The RA decoder aims to expand the input feature maps to higher resolution representations. This up-scaling is usually achieved through multiple deconvolution layers whose output is combined with previous feature maps to preserve spatial details. In our case, the representation is unusual due to the physical nature of the axes: The dimensions of the input tensor correspond respectively to range, Doppler and azimuth angle, whereas the feature maps that will be sent to the subsequent task heads should correspond to a Range-Azimuth representation. Consequently, we swap the Doppler and azimuth axes to match the final axis ordering and then upscale the feature maps.

However, the range axis has a lower size compared to the azimuth one, since it was decimated by a factor of 2 after each residual block, while the azimuth axis (formerly the channel axis) was increasing. Prior to these operations, we apply a 1×1 convolution to the feature maps from the encoder to the decoder. It adjusts the dimension of the azimuth channel to its final size, right before swapping the axes. The deconvolution layers upscale only the range axis, producing feature maps that are concatenated with those from the previous pyramid level. A final block of two Conv-BatchNorm-ReLU layers is applied, generating the final Range-Azimuth latent representation. The proposed RA decoder is illustrated in Figure 6.2.

6.3.4 Multi-task learning

Detection task. The detection head is inspired from PIXOR [Yang *et al.* 2018], an efficient and scalable single-stage model. Further details on the PIXOR architecture and method are provided in Section 2.6.4. It takes the RA latent representation as input and processes it using a first common sequence of four Conv-BatchNorm layers with 144, 96, 96 and 96 filters respectively. The branch is then divided in a classification and a regression heads. The classification part is a convolution layer with sigmoid activation that predicts a probability map. This output corresponds to a binary classification of each "pixel" as occupied or not by a vehicle. In order to reduce computational complexity, it predicts a coarse

RA map, where each cell has a resolution of 0.8m in range and $0.8°$ in azimuth (*i.e.*, $1/4$ and $1/8$ of native resolutions resp. in range and azimuth). This cell size is enough to dissociate two close objects. Then, the regression part finely predicts the range and azimuth values corresponding to the detected object. To do so, a unique 3×3 convolution layer outputs two feature maps corresponding to the final range and azimuth values.

This two-fold detection head is trained with a multi-task loss composed of a focal loss, specialized in unbalanced classification, applied to all the locations and of a "smooth L1" loss for the regression applied only on positive detection as detailed in the work of [Yang *et al.* 2018]. Let $\mathbf{x}$ be a training example, $\mathbf{y}_{\text{clas}} \in \{0,1\}^{B_R/4\times B_A/8}$ its classification ground truth and $\mathbf{y}_{\text{reg}} \in \mathbb{R}^{2\times B_R/4\times B_A/8}$ the associated regression ground truth. The detection head of FFT-RadNet predicts a detection map $\hat{\mathbf{y}}_{\text{clas}} \in [0,1]^{B_R/4\times B_A/8}$ and associated regression map $\hat{\mathbf{y}}_{\text{reg}} \in \mathbb{R}^{2\times B_R/4\times B_A/8}$. Its training loss is written:

$$\mathcal{L}_{\text{det}}(\mathbf{x}, \mathbf{y}_{\text{clas}}, \mathbf{y}_{\text{reg}}) = \text{focal}(\mathbf{y}_{\text{clas}}, \hat{\mathbf{y}}_{\text{clas}}) + \beta\, \text{smooth-L1}(\mathbf{y}_{\text{reg}} - \hat{\mathbf{y}}_{\text{reg}}), \tag{6.1}$$

where $\beta > 0$ is a balancing hyper-parameter. The focal and smooth-L1 are respectively defined as:

$$\text{focal}(\mathbf{y}, \mathbf{p}) = \begin{cases} -(1-\mathbf{p})^{\gamma} log(\mathbf{p}) & \text{if } \mathbf{y} = 1, \\ -\mathbf{p}^{\gamma} log(1-\mathbf{p}) & \text{otherwise,} \end{cases} \tag{6.2}$$

where γ is a down-weighting factor and

$$\text{smooth-L1}(\mathbf{x}) = \begin{cases} 0.5\mathbf{x}^2 & \text{if } \text{abs}\,\mathbf{x} < 1, \\ |\mathbf{x}| - 0.5 & \text{otherwise,} \end{cases} \tag{6.3}$$

where $|.|$ denotes the absolute value.

Segmentation task. The free driving space segmentation task is formulated as a pixel-level binary classification. The segmentation mask has a resolution of 0.4m in range and $0.2°$ in azimuth. It corresponds to half of the native range and azimuth resolutions while considering only half of the entire azimuth FoV (within $[-45°, 45°]$). The RA latent representation is processed by two consecutive groups of two Conv-BatchNorm-ReLu blocks, producing respectively 128 and 64 feature maps. A final 1×1 convolution outputs a 2D feature map followed by a sigmoid activation to estimate the probability of each location to be drivable. Let $\mathbf{x}$ be a training example, $\mathbf{y}_{\text{seg}} \in \{0,1\}^{B_R/2\times B_A/4}$ its one-hot ground truth and $\hat{\mathbf{y}}_{\text{seg}} \in [0,1]^{B_R/2\times B_A/4}$ the predicted soft detection map. The segmentation task is learnt using a BCE loss:

$$\mathcal{L}_{\text{free}}(\mathbf{x}, \mathbf{y}_{\text{seg}}) = \sum_{(r,a)\in\Omega} \text{BCE}(\mathbf{y}_{\text{seg}}(r,a), \hat{\mathbf{y}}_{\text{seg}}(r,a)), \tag{6.4}$$

where $\Omega = [\![1, \frac{B_R}{2}]\!] \times [\![1, \frac{B_A}{4}]\!]$.

Model	RADAR Input	Overall				Easy				Hard			
		AP(%) ↑	AR(%) ↑	R(cm) ↓	A(°) ↓	AP(%) ↑	AR(%) ↑	R(cm) ↓	A(°) ↓	AP(%) ↑	AR(%) ↑	R(cm) ↓	A(°) ↓
PIXOR	PC	96.46	32.32	0.17	0.25	**99.02**	28.83	0.15	0.19	93.28	38.69	0.19	0.33
PIXOR	RA	96.56	81.68	**0.10**	0.20	96.86	88.02	**0.09**	0.16	**95.88**	**70.10**	**0.12**	0.27
FFT-RadNet (ours)	RD	**96.84**	**82.18**	0.11	**0.17**	98.49	**91.69**	0.10	**0.13**	92.93	64.82	0.13	**0.26**

Table 6.3: **Object detection performances on RADIal Test split**. Comparison between PIXOR [Yang *et al.* 2018] trained with Point Cloud (PC) or Range-Azimuth (RA) representations, and the proposed FFT-RadNet requiring only RD as input. Our method obtains similar or better overall performances than baselines in both Average Precision (AP) and Average Recall (AP) for a 50% IoU threshold. It also reaches similar or better Range (R) and Angle (A) accuracy, showing it successfully learns a signal processing pipeline that estimates the AoA with significantly fewer operations, as detailed in Table 6.5.

End-to-end multi-task training. The whole FFT-RadNet model is trained by minimizing a combination of the previous detection and segmentation losses:

$$\mathcal{L}_{\text{MTL}} = \sum_{\mathbf{x}} \mathcal{L}_{\text{det}}(\mathbf{x}, \mathbf{y}_{\text{clas}}, \mathbf{y}_{\text{reg}}) + \lambda \mathcal{L}_{\text{free}}(\mathbf{x}, \mathbf{y}_{\text{seg}}), \tag{6.5}$$

w.r.t. the parameters of the MIMO pre-encoder, of the FPN encoder, of the RA decoder and of the two heads, where λ is a positive hyper-parameter that balances the two tasks. The following section will present the experiments and results of the FFT-RadNet with the RADIal dataset.

6.4 Experiments and Results

6.4.1 Training details

The proposed architecture has been trained on the RADIal dataset using exclusively the RD representations as input. The RD being composed of complex numbers, their real and imaginary parts are stacked along the channel axis and used as input of the MIMO pre-encoder. The dataset has been split into Training, Validation and Test sets (approximately 70%, 15% and 15% of the dataset, respectively) in such a way that frames from a same sequence can not appear in different sets. We manually split the Test dataset into "hard" and "easy" cases. Hard cases are mostly situations where the RADAR signal is perturbed, *e.g.*, by interference with other RADARs, important side-lobes effects or significant reflections on metallic surfaces.

The FFT-RadNet architecture is trained using the multi-task loss detailed in Section 6.3.4 with the following hyper-parameters set-up empirically: $\lambda = 100$, $\beta = 100$ and $\gamma = 2$. The training process uses the Adam optimizer [Kingma 2015] during 100 epochs, with an initial learning rate of 10^{-4} and a decay of 0.9 every 10 epochs.

6.4.2 Baselines

The proposed architecture has been compared to recent contributions in deep learning and in RADAR scene understanding. Most of the competing methods presented in Chapter

3 have been designed for LD RADAR and can not scale with HD RADAR data due to memory limitation. Instead, baselines with similar complexity have been selected regarding their input representation (Range-Azimuth or point cloud) for a fair comparison. Input representations (RD, RA or point cloud) are generated for the entire Training, Validation and Test sets using a conventional signal processing pipeline.

Object detection with point cloud. The PIXOR [Yang *et al.* 2018] method has been adapted to detect vehicles after voxelization of the RADAR point cloud into a 3D volume of $[0\,\text{m}, 103\,\text{m}]\times[-40\,\text{m}, 40\,\text{m}]\times[-2.5\,\text{m}, 2.0\,\text{m}]$ around the RADAR (longitudinal, lateral and vertical ranges), sampled at 0.1m in each direction. The size for this input 3D grid is thus $800\times1030\times45$. PIXOR is a lightweight architecture intended to be real-time. However, its input representation generates 96MB of data, which becomes a challenge for embedded devices.

Object detection with RA tensor. As detailed in Chapter 3 and Section 5.1, several methods [Major *et al.* 2019, Gao *et al.* 2020] including ours used views of the RAD tensor as input. However, the memory usage would be too extensive for HD RADAR data. As [Major *et al.* 2019] showed that using only the RA view leads to better performance for object detection, we compared our method to a PIXOR architecture without the voxelization module. It takes as input the RA representation in RADIal, of size 512×896 with range values in $[0\text{m}, 103\text{m}]$ and azimuth in $[-90°, 90°]$.

Freespace segmentation. We selected PolarNet [Nowruzi *et al.* 2020] to evaluate against our approach. It is a lightweight architecture designed to process RA maps and predict free space. We re-implemented the model to the best of our ability.

6.4.3 Evaluation metric

For object detection, the AP and AR are used considering an IoU threshold of 50%. For semantic segmentation, the mIoU metric is used on a binary classification task (*free* or *occupied*). The metric is computed on a reduced $[0\text{m}, 50\text{m}]$ range as the boundaries of the road surface are hardly visible beyond this distance. The evaluation metrics are detailed in Section 2.6.2.

6.4.4 Performance analysis

Object detection. Performances for object detection are reported in Table 6.4. We observe that FFT-RadNet using RD as input outperforms all baselines overall. The position accuracy, both in range and azimuth angle, is similar, and even better in angle, compared to PIXOR using RA as input (PIXOR-RA). These results show that our approach successfully learns the azimuth angle from the data. From a manufacturing viewpoint, note that this opens cost saving opportunities as the end-of-line calibration of the sensor is no longer required in the proposed framework. In the Easy Test set, FFT-RadNet delivers +1.6% AP and +3.6% AR compared to PIXOR-RA. However, on the Hard test set, PIXOR-RA

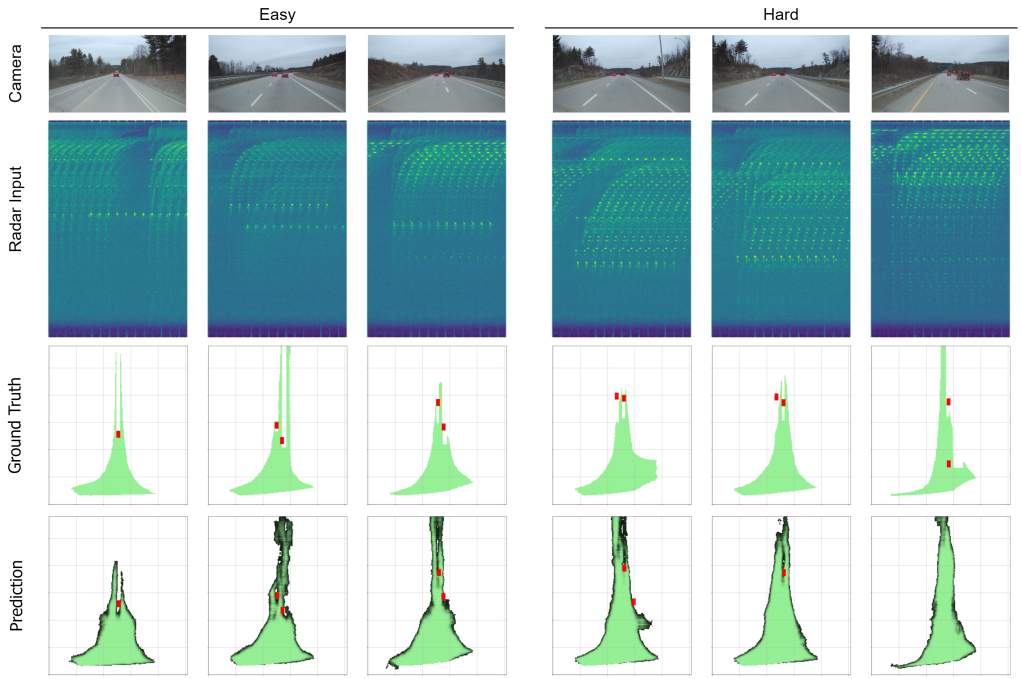


Figure 6.4: **Qualitative results for object detection and free space segmentation on Easy and Hard samples**. Camera views (1st row) are displayed for visual reference only; Range-Doppler representation (2nd row) are the only inputs to the model; Ground truths (3rd row) and predictions (4th row) are shown for both tasks. Ground truths and predictions are represented by Bird's Eye View maps with 2D red boxes for object detection and green shapes for free driving space segmentation. Note that there could be a projection error of the free driving space from camera to real world due to vehicle pitch variations.

performs the best. The RA approach works well with the hard samples because the data is pre-processed by a signal processing pipeline that already solves some of these cases. In contrast, the performance with point-cloud input is much lower than all others. Indeed, the recall is low due to the limited number of points at far range. Qualitative results of FFT-RadNet on the Easy and Hard Test sets are illustrated in Figure 6.4.

Free driving space segmentation. The performance for the free driving space segmentation is provided in Table 6.4.5. We observe that FFT-RadNet significantly outperforms PolarNet by 13.4% IoU on average. This is partly explained by the lack of elevation information in the RA representation, an information that is present in the RD. Segmentation results on the Easy and Hard Test sets are also presented in Figure 6.4.

6.4.5 Complexity analysis

FFT-RadNet has been designed to compress of the signal processing chains that transform the ADC data into either a sparse point cloud or denser representations (RA or RAD), without compromising the richness of the signal. Because the input data remains quite large,

Model	RADAR input	mIoU (%) ↑		
		Overall	Easy	Hard
PolarNet	RA	60.6	61.9	57.4
FFT-RadNet	RD	**74.0**	**74.6**	**72.3**

Table 6.4: **Free driving space segmentation performances**. FFT-RadNet successfully approximates the angle information in the RADAR data while reaching better performance than PolarNet. Note that this performance is achieved by FFT-RadNet while simultaneously performing object detection, as our model is multi-task.

Method	Input size (MB) ↓	# Params. (10^6) ↓	Complexity (GFLOPS) ↓ AoA processing	Model
PCL PIXOR	98.30	6.93	8	741
RA PIXOR	**1.75**	6.92	45*	761
FFT-RadNet	16.00	**3.79**	**0**	**584**

Table 6.5: **Complexity analysis of FFT-RadNet**. The proposed method reaches the best trade-off between the size of the input, the number of parameters of the model and the computational complexity. Note that the Angle-of-Arrival processing of the RA PIXOR method (*) considers only a single elevation, otherwise it is up to 496 GFLOPS for the whole set of $B_{\mathrm{E}} = 11$ elevations.

we designed a compact model to bound the complexity in terms of number of operations, as a trade-off between performance and range/angle accuracy. Moreover, the pre-encoder layer compresses significantly the input data. An ablation study has been performed to define the best trade-off between the size of the feature maps and the model's performance (details in Appendix C.1).

As shown in Table 6.5, FFT-RadNet is the only method that does not require the AoA estimation. As explained in Section 6.3.1, the pre-encoder layer compresses the MIMO signal containing all the information to recover both the azimuth and elevation angles. The AoA for the point cloud approach generates 3D coordinates for a sparse cloud of around 1000 points on average, leading to 8 GFLOPS worth of computing, prior to applying PIXOR for object detection.

To produce the RA or RAD tensor, the AoA runs for each single bin of the RD representation, but only considering one elevation. Such a model is thus unable to estimate the elevation of objects such as bridges or lost cargo (low object). For one elevation, the complexity is about 45 GFLOPS, but it would increase up to 495 GFLOPS for all the 11 elevations. We have demonstrated that FFT-RadNet can cut these processing costs without compromising the quality of the estimation.

6.5 Conclusions and discussions

This chapter introduced RADIal, a new dataset containing sequences of automotive-grade sensor signals (HD RADAR, camera and LiDAR). Synchronized sensor data are available in a raw format so that various representations can be evaluated and further research can be conducted, possibly with fusion-based approaches. The FFT-RadNet is also presented as a novel trainable architecture to process and analyse HD RADAR signals. We demonstrated that it effectively alleviates the need for costly pre-processing to estimate RA or RAD representations. Instead, it detects and estimates objects position while segmenting free driving space from the RD representations directly. Experiments on the RADIal dataset shown that FFT-RadNet slightly outperforms RA-based and point cloud-based approaches

while reducing processing requirements.

The annotations of the RADIal dataset have been performed semi-automatically and thus could be improved. Figure 6.4 illustrates misalignment of the camera's semantic segmentation mask projections to the Cartesian coordinates in the ground truths where the object boxes do not perfectly match with the shape of the free space segmentation. An additional correction could be integrated by quantifying the projection error due to the calibration of the camera. The azimuth angle estimation could be improved by defining additional pyramid features in the MIMO pre-encoder layer of the FFT-RadNet architecture. The number of feature maps defined at each level of the pyramid will correspond to a smoother angle resolution at each scale of the input. Finally, an extensive ablation study of the MIMO pre-encoder could also be explored to better understand the contribution of the virtual pairs of antennas regarding their position in the array of antennas in the sensor. Our experiments showed that best results are reached by considering a subset of the virtual antennas. A deeper analysis on which antennas contribute the most could be interesting to selectively build the array of antennas of a HD RADAR and possibly reducing its cost.

HD RADAR is expected to be increasingly used in the near feature for scene understanding due to its improved angular resolution. It is complementary to the LiDAR and camera sensor thanks to its robustness to adverse weather conditions and the recorded Doppler information. It could even replace the LiDAR sensor thanks to the azimuth and elevation resolutions achieved with hundreds of virtual antennas. Sensor fusion considering HD RADAR, LiDAR and camera will be explored to further improve scene understanding tasks in the context of autonomous driving.

CHAPTER 7

Conclusion

Contents

7.1 Contributions 125
7.2 Future work 127

7.1 Contributions

Advanced driving assistance systems and autonomous driving require a detailed perception around the ego vehicle. Robust scene understanding is key to ensuring safety of road users. Thus, complementary sensors recording redundant information are required to compensate for their limitations while taking advantage of their own physical properties. Since the recent golden age of deep learning, the performance of algorithms for scene understanding has increased dramatically and autonomous driving has become within reach. The RADAR sensor has been left behind due to its cumbersome and noisy representations which are complex to understand for human eyes. However, it is the only sensor that is robust to adverse weather conditions, while proving information on the velocity and position of the surrounding objects.

This thesis explored RADAR scene understanding using deep learning algorithms. We have addressed the fundamental problem of the lack of annotated datasets while proposing appropriate algorithms to explore RADAR representations. First, we created a simple simulation of RD representation by taking into account the properties of moving objects while including RADAR sensor noise (Section 4.1). We experimented with generative approaches to reconstruct RD representations from camera images with the underlying idea to automatically transfer annotations between the two domains (Section 4.2). Both methods aimed to build annotated RADAR dataset without the costly intervention of human experts. These methods have not been able to produce sufficiently realistic RADAR data to investigate these directions further. Therefore we proposed the CARRADA dataset (Section 4.3), composed of synchronised camera and annotated RADAR data for scene understanding. As detailed in Table 3.1, it is still the only dataset providing raw and processed RADAR data with both object detection and semantic segmentation annotations. A semi-automatic annotation pipeline has also been proposed, taking advantage of camera images to reduce the cost of manual annotation. Our annotation method has been slightly improved in the work of [Zhang *et al.* 2021a], as detailed in Section 5.1.4.1, demonstrating the relevance of its application to complex urban scenes. However, the efficiency of the camera projections

in the RADAR data is based on strong assumptions, *e.g.* non-occluded objects or objects at close range, and leads to inaccuracies in the annotations. Our proposed propagation of the annotations in the RADAR sequence alleviates the dependence on the camera image, but it is also limited regarding the object signature tracking which remains difficult when small objects are close together. Using increased human supervision or a higher resolution sensor could solve these problems. Finally, our method is limited in the generation of the RA annotations because it does not consider the angular resolution of the sensor. Therefore the generated annotations do not cover entirely the objects' signature on the RA view. This could be mitigated by incorporating the angular resolution of the sensor into the shape extension method presented in Section 4.3.2.3.

In the past two years, the release of annotated datasets has opened up research for RADAR scene understanding using deep learning algorithms. As detailed in Chapter 3, mostly object detection has been explored, however it is not well suited to the shapes of objects' signatures. We proposed a novel approach for multi-view RADAR semantic segmentation exploiting the raw RAD tensor of RADAR data while reducing its cumbersome representation and its noise (Section 5.1). Our proposed approach reached the state of the art performances of semantic segmentation considering several competing methods while requiring significantly fewer parameters. These results were confirmed after various ablation studies and evaluations on several datasets, including simple and complex urban scenes. Nonetheless, these small datasets are unbalanced and it is still difficult to provide a fair and precise evaluation on vulnerable road users (pedestrians and cyclists), which are usually under represented. Another limitation of our work is the aggregation method we employed for the RAD tensor. It helps to reduce the noise, as detailed in Section 2.2, but it also attenuates the high reflections of the objects making difficult to separate them. Moreover, the proposed Coherence Loss in Section 5.1.2.4 showed limits in improving radar semantic segmentation performances only on a single view. The spatial coherence could integrate the ground truth segmentation to penalize the network prediction in a robust manner.

Each sensor used for automotive scene understanding has its strengths and weaknesses. Sensor fusion aims to benefit from their advantages while compensating their limitations. In this thesis, we proposed a preliminary work to exploit jointly RADAR and LiDAR point clouds with an early fusion method (Section 5.2). It aims to quantify the resolution and the accuracy of a low-definition RADAR to construct a sensor uncertainty area for each measurement. The RADAR information of a point is then propagated to the LiDAR points belonging to its uncertainty area and vice-versa. The presented qualitative results showed that the RADAR information (Doppler and RCS) are well-propagated locally. The final point cloud benefits from the density of the LiDAR point cloud while carrying the Doppler and RCS information in dense local groups of points. The presented method is limited by the ghost and multi-path reflections of the RADAR point cloud propagating information in the wrong local space. However it can be tackled by filtering the upstream points. The RADAR and LiDAR point cloud fusion aims to improve scene understanding in general by exploiting both sensors at the same time. Various opportunities are discussed in the following section.

Our recent collaborative project (Chapter 6) explored the HD RADAR sensor by

proposing the RADIal dataset, the first dataset composed of camera, LiDAR and raw HD RADAR data annotated for object detection and free driving space segmentation. A deep neural network architecture is also proposed to directly learn from the raw RADAR and replace the costly pre-processing steps for angle estimation. It succeeds in attaining better performances than competing methods while requiring fewer parameters and operations. It also operates directly on the raw data, which is an important advantage for real-time applications using HD RADAR. The proposed dataset is an important step for RADAR understanding, but it has only a single class for object detection and the annotations are performed semi-automatically based on camera calibration, which is inaccurate and could be manually corrected. The proposed pre-encoder layer could also be improved by considering a smoother estimation of the angular resolution of the sensor which increases with the distance. Finally, a deeper study on the impact of the selected pairs of virtual antennas should be realized to better characterize the impact of each one of them. Since we reached the best performances by estimating fewer pairs of virtual antennas than existing in the sensor, such analysis should help to improve the geometric positioning of each antenna in an array of a RADAR.

7.2 Future work

Multiple datasets with RADAR data have been created in the past two years (see Table 3.1) opening up research in object detection and semantic segmentation for scene understanding. However, the community is still missing a large scale dataset containing camera, LiDAR and raw RADAR data with various scenarios (lighting and weather conditions) and diverse annotations. It is important to obtain such datasets to further explore fully supervised learning using RADAR data that the car will rely on in certain scenarios. It is also important to provide the raw data to identify the most appropriate representation in different driving scenarios. *E.g.*, considering LD RADAR, a point cloud representation is enough to detect vehicles but it will miss pedestrians at middle and long range.

We discussed in Chapter 5 the importance of raw data, in particular the RAD tensor, with semantic segmentation annotation for scene understanding using LD RADAR sensor. A simple aggregation method (see Equation 2.19) is then applied to reduce the noise and the size of the representation. The downside of this method is that high reflections are attenuated. In a current work, we are exploring diverse aggregation methods to further reduce the noise while respecting the high reflection distributions and separating them from the different classes. One way to carry out this idea is to use a weighted average depending on the bin intensity values. We note that the selection of specific slices per view could also be considered with a measure of signal disparity. We are also formulating the Coherence Loss differently to better balance the performance in multi-view RADAR semantic segmentation (see Section 5.1.2.4). In particular, we are trying to integrate the ground truth in the loss to better impose the spatial coherence between the prediction in the two branches of the neural network architecture. Since this loss aims to improve the performances in the RA view, we are also experimenting with a local back-propagation of the Coherence loss in specific branches of the network to avoid penalization in the RD branches.

In our opinion, raw data is the most relevant for scene understanding, but most of the available datasets and integrated algorithms use lightweight RADAR point clouds. This representation is sparse and misses numerous objects, but it carries the Doppler and the signal reflection (RCS). We proposed a method which propagates these information through the dense LiDAR point cloud while fusing them to benefit from their respective properties (Section 5.2). Our next work will consist in training deep neural network architectures specialized in point clouds (*e.g.* PointNet [Qi *et al.* 2017a] or PointNet++ [Qi *et al.* 2017b]) using our enriched point cloud to improve scene understanding tasks. Motivated by the lack of annotations and the difficulty of creating them even for a human, we will explore self-supervised learning consisting in training a model with a pretext task and to fine-tune it on a downstream application. As a first experiment, we will used our enriched point cloud to learn a model to predict the propagated Doppler components. This way, the model will be able to learn geometric features while distinguishing moving and static objects without any annotation cost. Our propagation and fusion module could be useful to consider RADAR data as prior weak annotations in a self-supervised setting. The main advantage is that it could be applied to any dataset with a LiDAR and at least one RADAR, either LD or HD.

The recent HD RADAR sensors attain a resolution similar to a LiDAR in azimuth and elevation angles but this requires costly pre-processing steps. This issue is barely discussed in the related work on HD RADAR but it is essential for real time predictions. In our collaborative work (Chapter 6), we showed that a neural network architecture is able to estimate the RADAR pre-processing while decreasing its computational complexity. This multi-task architecture also increases performance compared to recent competing methods, as each method is specialised in its own task. Real time predictions and high performances are reached with a HD RADAR. Moreover it provides representations as dense as LiDAR does with a comparable resolution. Additional experiments must be conducted to compare HD RADAR and LiDAR sensors performance individually on automotive scene understanding tasks. The HD RADAR suffers from ghost and multi-path reflections but if a neural network can distinguish the noise, as it does in our experiments, one can argue that this sensor may replace LiDAR in autonomous cars.

With respect to the limitations of the camera and LiDAR sensors, it is clear that the RADAR must be integrated into autonomous vehicles in order to obtain sufficient performance in any scenario to ensure the safety of drivers. In the near future, supervised learning using a RADAR sensor will be explored in depth to be able to rely solely on this sensor in several scenarios. Methods for sensor fusion using RADAR are at their early stage but they will be explored with any sensor mounted on a car. In particular, LD RADAR and LiDAR are particularly well suited to be fused and benefit from each of them at lower cost. Future research will also focus on HD RADAR for fusion, either with camera or LiDAR, to provide a dense representation of the scene surrounding the vehicle while benefiting from the properties of each one of them.

Scene understanding for autonomous driving is increasingly being mastered by exploiting multiple sensors in various driving scenarios. It goes hand in hand with decision making that uses the observation data to drive the car. The automotive industry estimates that vehicles with a level 3 of driving automation, *i.e.* full capacity to perform driving tasks

but requiring human intervention, will be sold to the public in 2024 in European countries. If research in this area continues to progress so rapidly, we can estimate that a level of automation that no longer requires a human driver could be available to the public within one or two decades.

Bibliography

[Abramowitz & Stegun 1965] Milton Abramowitz and Irene A. Stegun, editors. Handbook of mathematical functions: with formulas, graphs, and mathematical tables. Dover books on mathematics, 1965. (Cited on page 13.)

[Aydogdu *et al.* 2020] Cem Yusuf Aydogdu, Souvik Hazra, Avik Santra and Robert Weigel. *Multi-Modal Cross Learning for Improved People Counting using Short-Range FMCW Radar*. In IEEE International Radar Conference (RADAR), 2020. (Cited on page 43.)

[Azam *et al.* 2021] Shoaib Azam, Farzeen Munir and Moongu Jeon. *Channel Boosting Feature Ensemble for Radar-based Object Detection*. In IEEE Intelligent Vehicles Symposium (IV), 2021. (Cited on page 48.)

[Badrinarayanan *et al.* 2017] Vijay Badrinarayanan, Alex Kendall and Roberto Cipolla. *SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation*. In IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI), 2017. (Cited on page 50.)

[Bai *et al.* 2020] Jie Bai, Yifan Zhang, Libo Huang and Sen Li. *Vehicle Detection Based on Deep Neural Network Combined with Radar Attention Mechanism*. In Automotive Technical Papers, 2020. (Cited on page 52.)

[Barnes *et al.* 2020] Dan Barnes, Matthew Gadd, Paul Murcutt, Paul Newman and Ingmar Posner. *The Oxford Radar RobotCar Dataset: A Radar Extension to the Oxford RobotCar Dataset*. In IEEE International Conference on Robotics and Automation (ICRA), 2020. (Cited on page 45.)

[Bewley *et al.* 2016] Alex Bewley, Zongyuan Ge, Lionel Ott, Fabio Ramos and Ben Upcroft. *Simple Online and Realtime Tracking*. In IEEE International Conference on Image Processing (ICIP), 2016. (Cited on page 72.)

[Bijelic *et al.* 2018] Mario Bijelic, Tobias Gruber and Werner Ritter. *A Benchmark for Lidar Sensors in Fog: Is Detection Breaking Down?* In IEEE Intelligent Vehicles Symposium (IV), 2018. (Cited on pages 4 and 102.)

[Bijelic *et al.* 2020] Mario Bijelic, Tobias Gruber, Fahim Mannan, Florian Kraus, Werner Ritter, Klaus Dietmayer and Felix Heide. *Seeing Through Fog Without Seeing Fog: Deep Multimodal Sensor Fusion in Unseen Adverse Weather*. In IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2020. (Cited on page 55.)

[Brodeski *et al.* 2019] Daniel Brodeski, Igal Bilik and Raja Giryes. *Deep Radar Detector*. In IEEE Radar Conference (RadarConf), 2019. (Cited on page 48.)

[Brooker 2005] Graham Brooker. *Understanding millimetre wave FMCW radars*. In IEEE International Conference on Software Testing (ICST), 2005. (Cited on page 8.)

[Brooks *et al.* 2018] Daniel A. Brooks, Olivier Schwander, Frederic Barbaresco, Jean-Yves Schneider and Matthieu Cord. *Temporal Deep Learning for Drone Micro-Doppler Classification*. In International Radiation Symposium (IRS), 2018. (Cited on page 43.)

[Bugeau & Pérez 2007] Aurélie Bugeau and Patrick Pérez. *Bandwidth selection for kernel estimation in mixed multi-dimensional spaces*. Technical report RR-6286, INRIA, 2007. (Cited on page 75.)

[Caesar *et al.* 2020] Holger Caesar, Varun Bankiti, Alex H. Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan, Giancarlo Baldan and Oscar Beijbom. *nuScenes: A Multimodal Dataset for Autonomous Driving*. In IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2020. (Cited on pages 44, 102, 106, 168 and 172.)

[Capobianco *et al.* 2017] Samuele Capobianco, Luca Facheris, Fabrizio Cuccoli and Simone Marinai. *Vehicle classification based on convolutional networks applied to FM-CW radar signals*. In European Conference on Traffic Mining Applied to Police Activities (TRAP), 2017. (Cited on page 46.)

[Cennamo *et al.* 2020] Alessandro Cennamo, Florian Kaestner and Anton Kummert. *Leveraging Radar Features to Improve Point Clouds Segmentation with Neural Networks*. In International Conference on Engineering Applications of Neural Networks (EANN), 2020. (Cited on page 50.)

[Chadwick *et al.* 2019] Simon Chadwick, Will Maddern and Paul Newman. *Distant Vehicle Detection Using Radar and Vision*. In IEEE International Conference on Robotics and Automation (ICRA), 2019. (Cited on page 52.)

[Chen *et al.* 2017] Liang-Chieh Chen, George Papandreou, Florian Schroff and Hartwig Adam. *Rethinking Atrous Convolution for Semantic Image Segmentation*. In ArXiv, 2017. (Cited on page 37.)

[Chen *et al.* 2018a] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy and Alan L. Yuille. *DeepLab: Semantic Image Segmentation with Deep Convolutional Nets, Atrous Convolution, and Fully Connected CRFs*. In IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI), 2018. (Cited on pages 37, 87, 89, 99 and 162.)

[Chen *et al.* 2018b] Liang-Chieh Chen, Yukun Zhu, George Papandreou, Florian Schroff and Hartwig Adam. *Encoder-Decoder with Atrous Separable Convolution for Semantic Image Segmentation*. In European Conference on Computer Vision (ECCV), 2018. (Cited on pages 37, 50, 87, 89, 94, 114 and 165.)

[Chen *et al.* 2021a] Jinbo Chen, Dongheng Zhang, Zhi Wu, Fang Zhou, Qibin Sun and Yan Chen. *Contactless Electrocardiogram Monitoring with Millimeter Wave Radar*. In ArXiv, 2021. (Cited on page 43.)

[Chen *et al.* 2021b] Shiliang Chen, Wentao He, Jianfeng Ren and Xudong Jiang. *Attention-based Dual-stream Vision Transformer for Radar Gait Recognition*. In ArXiv, 2021. (Cited on page 42.)

[Cheng & Liu 2021] Yuwei Cheng and Yimin Liu. *Person Reidentification Based on Automotive Radar Point Clouds*. In IEEE Transactions on Geoscience and Remote Sensing, 2021. (Cited on page 46.)

[Cheng *et al.* 2021a] Yuwei Cheng, Jingran Su, Hongyu Chen and Yimin Liu. *A New Automotive Radar 4D Point Clouds Detector by Using Deep Learning*. In IEEE International Conference on Acoustics, Speech, & Signal Processing (ICASSP), 2021. (Cited on page 50.)

[Cheng *et al.* 2021b] Yuwei Cheng, Hu Xu and Yimin Liu. *Robust Small Object Detection on the Water Surface Through Fusion of Camera and Millimeter Wave Radar*. In IEEE International Conference on Computer Vision (ICCV), 2021. (Cited on page 52.)

[Chollet 2017] François Chollet. *Xception: Deep Learning with Depthwise Separable Convolutions*. In IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017. (Cited on page 37.)

[Comaniciu & Meer 2002] D. Comaniciu and P. Meer. *Mean shift: a robust approach toward feature space analysis*. In IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI), 2002. (Cited on pages 50 and 75.)

[Cordts *et al.* 2016] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth and Bernt Schiele. *The Cityscapes Dataset for Semantic Urban Scene Understanding*. In IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016. (Cited on pages 5 and 34.)

[Cui *et al.* 2021] Hang Cui, Junzhe Wu, Jiaming Zhang, Girish Chowdhary and William R. Norris. *3D Detection and Tracking for On-road Vehicles with a Monovision Camera and Dual Low-cost 4D mmWave Radars*. In International Conference on Intelligent Transportation Systems (ITSC), 2021. (Cited on page 52.)

[Dai *et al.* 2016] Jifeng Dai, Yi Li, Kaiming He and Jian Sun. *R-FCN: Object Detection via Region-Based Fully Convolutional Networks*. In Conference on Neural Information Processing Systems (NeurIPS), 2016. (Cited on page 157.)

[Dai *et al.* 2017] Jifeng Dai, Haozhi Qi, Yuwen Xiong, Yi Li, Guodong Zhang, Han Hu and Yichen Wei. *Deformable Convolutional Networks*. In IEEE International Conference on Computer Vision (ICCV), 2017. (Cited on page 42.)

[Dalsasso *et al.* 2021] Emanuele Dalsasso, Loïc Denis and Florence Tupin. *SAR2SAR: a semi-supervised despeckling algorithm for SAR images*. In Journal of Selected Topics in Applied Earth Observations and Remote Sensing, 2021. (Cited on pages 12 and 13.)

[Danzer *et al.* 2019] Andreas Danzer, Thomas Griebel, Martin Bach and Klaus Dietmayer. *2D Car Detection in Radar Data with PointNets*. In International Conference on Intelligent Transportation Systems (ITSC), 2019. (Cited on page 48.)

[de Oliveira & Bekooij 2020] Marcio L. Lima de Oliveira and Marco J. G. Bekooij. *Deep Convolutional Autoencoder Applied for Noise Reduction in Range-Doppler Maps of FMCW Radars*. In IEEE International Radar Conference (RADAR), 2020. (Cited on page 43.)

[Dekker *et al.* 2017] B. Dekker, S. Jacobs, A. S. Kossen, M. C. Kruithof, A. G. Huizing and M. Geurts. *Gesture recognition with a low power FMCW radar and a deep convolutional neural network*. In IEEE European Radar Conference (EuRAD), 2017. (Cited on page 42.)

[Deledalle *et al.* 2017] Charles-Alban Deledalle, Loic Denis, Sonia Tabti and Florence Tupin. *MuLoG, or How to Apply Gaussian Denoisers to Multi-Channel SAR Speckle Reduction?* In IEEE Transactions on Image Processing, 2017. (Cited on page 13.)

[Deng *et al.* 2009] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li and Li Fei-Fei. *Imagenet: A large-scale hierarchical image database*. In IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2009. (Cited on page 26.)

[Djuric *et al.* 2021] Nemanja Djuric, Henggang Cui, Zhaoen Su, Shangxuan Wu, Huahua Wang, Fang-Chieh Chou, Luisa San Martin, Song Feng, Rui Hu, Yang Xu, Alyssa Dayan, Sidney Zhang, Brian C. Becker, Gregory P. Meyer, Carlos Vallespi-Gonzalez and Carl K. Wellington. *MultiXNet: Multiclass Multistage Multimodal Motion Prediction*. In IEEE Intelligent Vehicles Symposium (IV), 2021. (Cited on page 55.)

[Dong *et al.* 2020] Xu Dong, Pengluo Wang, Pengyue Zhang and Langechuan Liu. *Probabilistic Oriented Object Detection in Automotive Radar*. In IEEE Conference on Computer Vision and Pattern Recognition Workshop (CVPRW), 2020. (Cited on page 47.)

[Dong *et al.* 2021] Xu Dong, Binnan Zhuang, Yunxiang Mao and Langechuan Liu. *Radar Camera Fusion via Representation Learning in Autonomous Driving*. In IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2021. (Cited on page 52.)

[Donnet & Longstaff 2006] B Donnet and I Longstaff. *MIMO Radar, Techniques and Opportunities*. In IEEE European Radar Conference (EuRAD), 2006. (Cited on pages 8 and 9.)

[Dosovitskiy *et al.* 2021] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit and Neil Houlsby. *An*

Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. In International Conference on Learning Representations (ICLR), 2021. (Cited on pages 29 and 42.)

[Endres & Schindelin 2003] D.M. Endres and J.E. Schindelin. *A new metric for probability distributions*. In IEEE Transactions on Information Theory, 2003. (Cited on page 77.)

[Ester *et al.* 1996] Martin Ester, Hans-Peter Kriegel, Jörg Sander and Xiaowei Xu. *A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise*. In Knowledge Discovery and Data Mining (KDD), 1996. (Cited on page 96.)

[Everingham *et al.* 2015] M. Everingham, S. M. A. Eslami, L. Van Gool, C. K. I. Williams, J. Winn and A. Zisserman. *The Pascal Visual Object Classes Challenge: A Retrospective*. In International Journal of Computer Vision (IJCV), 2015. (Cited on pages 29 and 34.)

[Farag 2021] Wael Farag. *Real-time lidar and radar fusion for road-objects detection and tracking*. In International Journal of Computational Science and Engineering (IJCSE), 2021. (Cited on page 54.)

[Feng *et al.* 2019] Zhaofei Feng, Shuo Zhang, Martin Kunert and Werner Wiesbeck. *Point Cloud Segmentation with a High-Resolution Automotive Radar*. In IEEE Automotive meets Electronics (AmE), 2019. (Cited on page 50.)

[Fu *et al.* 2018] Huan Fu, Mingming Gong, Chaohui Wang, Kayhan Batmanghelich and Dacheng Tao. *Deep Ordinal Regression Network for Monocular Depth Estimation*. In IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2018. (Cited on page 53.)

[Gadd *et al.* 2021] Matthew Gadd, Daniele De Martini and Paul Newman. *Unsupervised Place Recognition with Deep Embedding Learning over Radar Videos*. In IEEE International Conference on Robotics and Automation Workshop (ICRAW), 2021. (Cited on page 43.)

[Gao *et al.* 2019] Xiangyu Gao, Guanbin Xing, Sumit Roy and Hui Liu. *Experiments with mmWave Automotive Radar Test-bed*. In Asilomar Conference, 2019. (Cited on page 47.)

[Gao *et al.* 2020] Xiangyu Gao, Guanbin Xing, Sumit Roy and Hui Liu. *RAMP-CNN: A Novel Neural Network for Enhanced Automotive Radar Object Recognition*. In IEEE Sensors Journal, 2020. (Cited on pages 47, 87, 89 and 120.)

[Garcia *et al.* 2012] Fernando Garcia, Pietro Cerri, Alberto Broggi, Arturo de la Escalera and Jose Maria Armingol. *Data fusion for overtaking vehicle detection based on radar and optical flow*. In IEEE Intelligent Vehicles Symposium (IV), 2012. (Cited on page 51.)

[Garnot *et al.* 2021] Vivien Sainte Fare Garnot, Loic Landrieu and Nesrine Chehata. *Multi-Modal Temporal Attention Models for Crop Mapping from Satellite Time Series*. In ArXiv, 2021. (Cited on page 43.)

[Gasperini *et al.* 2021] Stefano Gasperini, Patrick Koch, Vinzenz Dallabetta, Nassir Navab, Benjamin Busam and Federico Tombari. *R4Dyn: Exploring Radar for Self-Supervised Monocular Depth Estimation of Dynamic Scenes*. In IEEE International Conference on 3D Vision (3DV), 2021. (Cited on page 52.)

[Geiger *et al.* 2013] A Geiger, P Lenz, C Stiller and R Urtasun. *Vision meets robotics: The KITTI dataset*. In International Journal of Robotics Research, 2013. (Cited on page 5.)

[Ghaleb 2009] Antoine Ghaleb. *Micro-Doppler analysis of non-stationary moving targets in radar imaging*. PhD thesis, Telecom Paris, France, 2009. (Cited on pages 8 and 11.)

[Girshick *et al.* 2016] Ross Girshick, Jeff Donahue, Trevor Darrell and Jitendra Malik. *Region-Based Convolutional Networks for Accurate Object Detection and Segmentation*. In IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI), 2016. (Cited on pages 30 and 36.)

[Girshick 2015] Ross Girshick. *Fast R-CNN*. In IEEE International Conference on Computer Vision (ICCV), September 2015. (Cited on pages 31 and 36.)

[Glorot & Bengio 2010] Xavier Glorot and Yoshua Bengio. *Understanding the difficulty of training deep feedforward neural networks*. In International Conference on Artificial Intelligence and Statistics (AISTATS), 2010. (Cited on pages 19 and 62.)

[Goodfellow *et al.* 2014] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville and Yoshua Bengio. *Generative Adversarial Networks*. In Conference on Neural Information Processing Systems (NeurIPS), 2014. (Cited on pages 43, 68 and 69.)

[Goodfellow *et al.* 2016] Ian J. Goodfellow, Yoshua Bengio and Aaron Courville. Deep Learning. MIT Press, 2016. (Cited on pages 19, 20, 24 and 26.)

[Goodman 1976] J. W. Goodman. *Some fundamental properties of speckle*. In Journal of the Optical Society of America, 1976. (Cited on pages 12 and 13.)

[Goodman 2007] Joseph W Goodman. Speckle phenomena in optics: theory and applications. Roberts and Company Publishers, 2007. (Cited on pages 59, 64 and 65.)

[Graham *et al.* 2018] Benjamin Graham, Martin Engelcke and Laurens van der Maaten. *3D Semantic Segmentation with Submanifold Sparse Convolutional Networks*. In IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2018. (Cited on page 55.)

[Graves *et al.* 2006] Alex Graves, Santiago Fernández, Faustino Gomez and Jürgen Schmidhuber. *Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks*. In International Conference on Machine Learning (ICML), 2006. (Cited on page 42.)

[Griebel *et al.* 2021] Thomas Griebel, Dominik Authaler, Markus Horn, Matti Henning, Michael Buchholz and Klaus Dietmayer. *Anomaly Detection in Radar Data Using PointNets*. In IEEE Intelligent Transportation Systems Conference (ITSC), 2021. (Cited on page 51.)

[Grimm *et al.* 2020] Christopher Grimm, Tai Fei, Ernst Warsitz, Ridha Farhoud, Tobias Breddermann and Reinhold Haeb-Umbach. *Warping of Radar Data into Camera Image for Cross-Modal Supervision in Automotive Applications*. In ArXiv, 2020. (Cited on page 55.)

[Guan *et al.* 2020] Junfeng Guan, Sohrab Madani, Suraj Jog, Saurabh Gupta and Haitham Hassanieh. *Through Fog High-Resolution Imaging Using Millimeter Wave Radar*. In IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2020. (Cited on pages 4 and 102.)

[Guo *et al.* 2020] Yulan Guo, Hanyun Wang, Qingyong Hu, Hao Liu, Li Liu and Mohammed Bennamoun. *Deep Learning for 3D Point Clouds: A Survey*. In IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI), 2020. (Cited on page 40.)

[Hastie *et al.* 2001] Trevor Hastie, Robert Tibshirani and Jerome Friedman. The Elements of Statistical Learning. Springer Series in Statistics, 2001. (Cited on page 15.)

[Hazra & Santra 2019] Souvik Hazra and Avik Santra. *Radar Gesture Recognition System in Presence of Interference using Self-Attention Neural Network*. In IEEE International Conference on Machine Learning and Applications (ICMLA), 2019. (Cited on page 42.)

[He *et al.* 2016] Kaiming He, Xiangyu Zhang, Shaoqing Ren and Jian Sun. *Deep residual learning for image recognition*. In IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016. (Cited on pages 4, 28, 33, 37, 62, 93, 117, 157, 159 and 160.)

[He *et al.* 2017] Kaiming He, Georgia Gkioxari, Piotr Dollar and Ross Girshick. *Mask R-CNN*. In IEEE International Conference on Computer Vision (ICCV), 2017. (Cited on pages 33, 36, 72 and 96.)

[Hearst *et al.* 1998] M.A. Hearst, S.T. Dumais, E. Osuna, J. Platt and B. Scholkopf. *Support vector machines*. In IEEE Intelligent Systems and their Applications, 1998. (Cited on page 30.)

[Hinton *et al.* 2012] Geoffrey E. Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever and Ruslan R. Salakhutdinov. *Improving neural networks by preventing co-adaptation of feature detectors*. In ArXiv, 2012. (Cited on page 21.)

[Hirschmuller 2008] H. Hirschmuller. *Stereo Processing by Semiglobal Matching and Mutual Information*. In IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI), 2008. (Cited on page 96.)

[Hochreiter & Schmidhuber 1997] Sepp Hochreiter and Jürgen Schmidhuber. *Long short-term memory*. In Neural Computation, 1997. (Cited on page 26.)

[Hoermann *et al.* 2018] Stefan Hoermann, Martin Bach and Klaus Dietmayer. *Dynamic Occupancy Grid Prediction for Urban Autonomous Driving: A Deep Learning Approach with Fully Automatic Labeling*. In IEEE International Conference on Robotics and Automation (ICRA), 2018. (Cited on page 50.)

[Hsu *et al.* 2021] Chih-Chung Hsu, Chieh Lee, Lin Chen, Min-Kai Hung, Yu-Lun Lin and Xian-Yu Wang. *Efficient-ROD: Efficient Radar Object Detection based on Densely Connected Residual Network*. In ACM International Conference on Multimedia Retrieval (ICMR), 2021. (Cited on page 47.)

[Hu *et al.* 2018] Jie Hu, Li Shen and Gang Sun. *Squeeze-and-Excitation Networks*. In IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2018. (Cited on pages 29 and 160.)

[Huang *et al.* 2019] Xinyu Huang, Peng Wang, Xinjing Cheng, Dingfu Zhou, Qichuan Geng and Ruigang Yang. *The ApolloScape Open Dataset for Autonomous Driving and its Application*. In IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI), 2019. (Cited on page 5.)

[Hussain *et al.* 2021] Muhamamd Ishfaq Hussain, Muhammad Aasim Rafique and Moongu Jeon. *RVMDE: Radar Validated Monocular Depth Estimation for Robotics*. In ArXiv, 2021. (Cited on page 53.)

[Ioffe & Szegedy 2015] Sergey Ioffe and Christian Szegedy. *Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift*. In International Conference on Machine Learning (ICML), 2015. (Cited on page 23.)

[Iovescu & Rao 2017] Cesar Iovescu and Sandeep Rao. *The fundamentals of millimeter wave radar sensors*. Technical report, Texas Instruments, 2017. (Cited on page 12.)

[Ishak *et al.* 2018] K. Ishak, N. Appenrodt, J. Dickmann and C. Waldschmidt. *Human Motion Training Data Generation for Radar Based Deep Learning Applications*. In IEEE International Conference on Microwaves for Intelligent Mobility, 2018. (Cited on page 42.)

[Isola *et al.* 2017] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou and Alexei A. Efros. *Image-to-Image Translation with Conditional Adversarial Networks*. In IEEE Conference

on Computer Vision and Pattern Recognition (CVPR), 2017. (Cited on pages 53, 68 and 69.)

[Janai *et al.* 2020] Joel Janai, Fatma Güney, Aseem Behl and Andreas Geiger. *Computer Vision for Autonomous Vehicles: Problems, Datasets and State of the Art*. In FNT in Computer Graphics and Vision, 2020. (Cited on page 1.)

[Ji *et al.* 2012] Shuiwang Ji, Wei Xu, Ming Yang and Kai Yu. *3D convolutional neural networks for human action recognition*. In IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI), 2012. (Cited on page 66.)

[Jin *et al.* 2021] Shaojie Jin, Ying Gao, Shoucai Jing, Fei Hui, Xiangmo Zhao and Jianzhen Liu. *Traffic Flow Parameters Collection under Variable Illumination Based on Data Fusion*. In Journal of Advanced Transportation, 2021. (Cited on page 52.)

[Ju *et al.* 2021] Bo Ju, Wei Yang, Jinrang Jia, Xiaoqing Ye, Qu Chen, Xiao Tan, Hao Sun, Yifeng Shi and Errui Ding. *DANet: Dimension Apart Network for Radar Object Detection*. In ACM International Conference on Multimedia Retrieval (ICMR), 2021. (Cited on page 48.)

[Karlsson *et al.* 2021] Robin Karlsson, David Robert Wong, Kazunari Kawabata, Simon Thompson and Naoki Sakai. *Probabilistic Rainfall Estimation from Automotive Lidar*. In ArXiv, 2021. (Cited on pages 4 and 102.)

[Karpathy 2021] Andrej Karpathy. *CS231n Convolutional Neural Networks for Visual Recognition.*, 2021. (Cited on pages 16, 18 and 24.)

[Kaul *et al.* 2020] Prannay Kaul, Daniele De Martini, Matthew Gadd and Paul Newman. *RSS-Net: Weakly-Supervised Multi-Class Semantic Segmentation with FMCW Radar*. In IEEE Intelligent Vehicles Symposium (IV), 2020. (Cited on pages 50, 87, 89, 90, 93 and 162.)

[Kiefer & Wolfowitz 1952] J. Kiefer and J. Wolfowitz. *Stochastic Estimation of the Maximum of a Regression Function*. In The Annals of Mathematical Statistics, 1952. (Cited on page 19.)

[Kim & Moon 2016] Youngwook Kim and Taesup Moon. *Human Detection and Activity Classification Based on Micro-Doppler Signatures Using Deep Convolutional Neural Networks*. In Geoscience and Remote Sensing Letters, 2016. (Cited on page 42.)

[Kim & Toomajian 2016] Y. Kim and B. Toomajian. *Hand Gesture Recognition Using Micro-Doppler Signatures With Convolutional Neural Network*. In IEEE Access, 2016. (Cited on page 42.)

[Kim *et al.* 2020a] Giseop Kim, Yeong Sang Park, Younghun Cho, Jinyong Jeong and Ayoung Kim. *MulRan: Multimodal Range Dataset for Urban Place Recognition*. In

IEEE International Conference on Robotics and Automation (ICRA), 2020. (Cited on page 45.)

[Kim *et al.* 2020b] Jinhyeong Kim, Youngseok Kim and Dongsuk Kum. *Low-level Sensor Fusion Network for 3D Vehicle Detection using Radar Range-Azimuth Heatmap and Monocular Image*. In Asian Conference on Computer Vision (ACCV), 2020. (Cited on page 53.)

[Kim *et al.* 2020c] Youngseok Kim, Jun Won Choi and Dongsuk Kum. *GRIF Net: Gated Region of Interest Fusion Network for Robust 3D Object Detection from Radar Point Cloud and Monocular Image*. In IEEE International Conference on Intelligent Robots and Systems (IROS), 2020. (Cited on page 53.)

[Kingma 2015] Diederik P. Kingma. *Adam: a method for stochastic optimization*. In International Conference on Learning Representations (ICLR), 2015. (Cited on pages 20, 62, 80, 93 and 119.)

[Klarenbeek *et al.* 2017] G. Klarenbeek, R. I. A. Harmanny and L. Cifola. *Multi-target human gait classification using LSTM recurrent neural networks applied to micro-Doppler*. In IEEE European Radar Conference (EuRAD), 2017. (Cited on page 42.)

[Kopp *et al.* 2021] Johannes Kopp, Dominik Kellner, Aldi Piroli and Klaus Dietmayer. *Fast Rule-Based Clutter Detection in Automotive Radar Data*. In International Conference on Intelligent Transportation Systems (ITSC), 2021. (Cited on page 108.)

[Kowol *et al.* 2021] Kamil Kowol, Matthias Rottmann, Stefan Bracke and Hanno Gottschalk. *YOdar: Uncertainty-based Sensor Fusion for Vehicle Detection with Camera and Radar Sensors*. In International Conference on Agents and Artificial Intelligence (ICAART), 2021. (Cited on page 51.)

[Krizhevsky *et al.* 2012] Alex Krizhevsky, Ilya Sutskever and Geoffrey E Hinton. *ImageNet Classification with Deep Convolutional Neural Networks*. In Conference on Neural Information Processing Systems (NeurIPS), 2012. (Cited on pages 4, 23 and 27.)

[Krähenbühl & Koltun 2011] Philipp Krähenbühl and Vladlen Koltun. *Efficient Inference in Fully Connected CRFs with Gaussian Edge Potentials*. In Conference on Neural Information Processing Systems (NeurIPS), 2011. (Cited on page 37.)

[Ku *et al.* 2018] Jason Ku, Melissa Mozifian, Jungwook Lee, Ali Harakeh and Steven Waslander. *Joint 3D Proposal Generation and Object Detection from View Aggregation*. In IEEE International Conference on Intelligent Robots and Systems (IROS), 2018. (Cited on page 52.)

[Kuang *et al.* 2020] Hongwu Kuang, Xiaodong Liu, Jingwei Zhang and Zicheng Fang. *Multi-Modality Cascaded Fusion Technology for Autonomous Driving*. In IEEE

International Conference on Robotics and Automation Sciences (ICRAS), 2020. (Cited on page 53.)

[Kullback & Leibler 1951] S. Kullback and R. A. Leibler. *On Information and Sufficiency*. In The Annals of Mathematical Statistics, 1951. (Cited on page 77.)

[Kung *et al.* 2021] Pou-Chun Kung, Chieh-Chih Wang and Wen-Chieh Lin. *Radar Occupancy Prediction with Lidar Supervision while Preserving Long-Range Sensing and Penetrating Capabilities*. In ArXiv, 2021. (Cited on page 54.)

[Kurup & Bos 2021] Akhil Kurup and Jeremy Bos. *DSOR: A Scalable Statistical Filter for Removing Falling Snow from LiDAR Point Clouds in Severe Winter Weather*. In ArXiv, 2021. (Cited on page 4.)

[LeCun *et al.* 1989] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard and L. D. Jackel. *Backpropagation Applied to Handwritten Zip Code Recognition*. In Neural Computation, 1989. (Cited on pages 4 and 22.)

[LeCun *et al.* 2012] Yann LeCun, Léon Bottou, Genevieve B. Orr and Klaus-Robert Müller. *Efficient BackProp*. In Neural Networks: Tricks of the Trade. Springer, 2012. (Cited on page 27.)

[Lee, Wei-Yu *et al.* 2021] Lee, Wei-Yu, Martin Dimitrievski, Ljubomir Jovanov and Wilfried Philips. *Spatio-temporal consistency for semi-supervised learning using 3D radar cubes*. In IEEE Intelligent Vehicles Symposium (IV), 2021. (Cited on page 48.)

[Lei *et al.* 2020] Wentai Lei, Xinyue Jiang, Long Xu, Jiabin Luo, Mengdi Xu and Feifei Hou. *Continuous Gesture Recognition Based on Time Sequence Fusion Using MIMO Radar Sensor and Deep Learning*. In Electronics, 2020. (Cited on page 42.)

[Lekic & Babic 2019] Vladimir Lekic and Zdenka Babic. *Automotive radar and camera fusion using Generative Adversarial Networks*. In Computer Vision and Image Understanding (CVIU), 2019. (Cited on page 53.)

[Li & Xie 2020] Liang-qun Li and Yuan-liang Xie. *A Feature Pyramid Fusion Detection Algorithm Based on Radar and Camera Sensor*. In IEEE International Conference on Intelligent Computing and Signal Processing (ICSP), 2020. (Cited on page 53.)

[Liang *et al.* 2020] Ming Liang, Bin Yang, Wenyuan Zeng, Yun Chen, Rui Hu, Sergio Casas and Raquel Urtasun. *PnPNet: End-to-End Perception and Prediction with Tracking in the Loop*. In IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2020. (Cited on page 54.)

[Lin *et al.* 2014a] Min Lin, Qiang Chen and Shuicheng Yan. *Network In Network*. In International Conference on Learning Representations (ICLR), 2014. (Cited on page 28.)

[Lin *et al.* 2014b] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár and C. Lawrence Zitnick. *Microsoft COCO: Common Objects in Context*. In European Conference on Computer Vision (ECCV), 2014. (Cited on pages 29 and 34.)

[Lin *et al.* 2017a] Tsung-Yi Lin, Piotr Dollar, Ross Girshick, Kaiming He, Bharath Hariharan and Serge Belongie. *Feature Pyramid Networks for Object Detection*. In IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017. (Cited on pages 36 and 37.)

[Lin *et al.* 2017b] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He and Piotr Dollár. *Focal Loss for Dense Object Detection*. In IEEE International Conference on Computer Vision (ICCV), 2017. (Cited on page 114.)

[Lin *et al.* 2020] Juan-Ting Lin, Dengxin Dai and Luc Van Gool. *Depth Estimation from Monocular Images and Sparse Radar Data*. In IEEE International Conference on Intelligent Robots and Systems (IROS), 2020. (Cited on page 53.)

[Liu *et al.* 2015] Wei Liu, Andrew Rabinovich and Alexander C. Berg. *ParseNet: Looking Wider to See Better*. In ArXiv, 2015. (Cited on page 158.)

[Liu *et al.* 2016] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott E. Reed, Cheng-Yang Fu and Alexander C. Berg. *SSD: Single Shot MultiBox Detector*. In European Conference on Computer Vision (ECCV), 2016. (Cited on pages 32 and 55.)

[Liu *et al.* 2018] Shu Liu, Lu Qi, Haifang Qin, Jianping Shi and Jiaya Jia. *Path Aggregation Network for Instance Segmentation*. In IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2018. (Cited on page 160.)

[Liu *et al.* 2021a] Jianan Liu, Weiyi Xiong, Liping Bai, Yuxuan Xia and Bing Zhu. *Deep Instance Segmentation with High-Resolution Automotive Radar*. In ArXiv, 2021. (Cited on page 51.)

[Liu *et al.* 2021b] Ze Liu, Yingfeng Cai, Hai Wang and Long Chen. *Surrounding Objects Detection and Tracking for Autonomous Driving Using LiDAR and Radar Fusion*. In Chinese Journal of Mechanical Engineering (CJME), 2021. (Cited on page 54.)

[Liu *et al.* 2021c] Ze Liu, Yingfeng Cai, Hai Wang, Long Chen, Hongbo Gao, Yunyi Jia and Yicheng Li. *Robust Target Recognition and Tracking of Self-Driving Cars With Radar and Camera Information Fusion Under Severe Weather Conditions*. In IEEE Transactions on Intelligent Transportation Systems, 2021. (Cited on page 52.)

[Liu *et al.* 2021d] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin and Baining Guo. *Swin Transformer: Hierarchical Vision Transformer using Shifted Windows*. In IEEE International Conference on Computer Vision (ICCV), 2021. (Cited on page 29.)

[Lo & Vandewalle 2021] Chen-Chou Lo and Patrick Vandewalle. *Depth Estimation from Monocular Images and Sparse radar using Deep Ordinal Regression Network*. In IEEE International Conference on Image Processing (ICIP), 2021. (Cited on page 53.)

[Lombacher *et al.* 2017] Jakob Lombacher, Kilian Laudt, Markus Hahn, Jurgen Dickmann and Christian Wohler. *Semantic radar grids*. In IEEE Intelligent Vehicles Symposium (IV), 2017. (Cited on page 50.)

[Long *et al.* 2015] Jonathan Long, Evan Shelhamer and Trevor Darrell. *Fully convolutional networks for semantic segmentation*. In IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2015. (Cited on pages 35, 50, 79, 87, 94, 99, 158 and 165.)

[Long *et al.* 2021a] Yunfei Long, Daniel Morris, Xiaoming Liu, Marcos Castro, Punarjay Chakravarty and Praveen Narayanan. *Full-Velocity Radar Returns by Radar-Camera Fusion*. In IEEE International Conference on Computer Vision (ICCV), 2021. (Cited on page 51.)

[Long *et al.* 2021b] Yunfei Long, Daniel Morris, Xiaoming Liu, Marcos Castro, Punarjay Chakravarty and Praveen Narayanan. *Radar-Camera Pixel Depth Association for Depth Completion*. In IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2021. (Cited on page 51.)

[Lowe 2004] David G. Lowe. *Distinctive Image Features from Scale-Invariant Keypoints*. In International Journal of Computer Vision (IJCV), 2004. (Cited on page 27.)

[Major *et al.* 2019] Bence Major, Daniel Fontijne, Amin Ansari, Ravi Teja Sukhavasi, Radhika Gowaikar, Michael Hamilton, Sean Lee, Slawomir Grzechnik and Sundar Subramanian. *Vehicle Detection With Automotive Radar Using Deep Learning on Range-Azimuth-Doppler Tensors*. In IEEE International Conference on Computer Vision Workshop (ICCVW), 2019. (Cited on pages 47, 81 and 120.)

[Masci *et al.* 2011] Jonathan Masci, Ueli Meier, Dan Cireşan and Jürgen Schmidhuber. *Stacked Convolutional Auto-Encoders for Hierarchical Feature Extraction*. In International Conference on Artificial Neural Networks (ICANN), 2011. (Cited on page 67.)

[McCulloch & Pitts 1943] Warren S. McCulloch and Walter Pitts. *A logical calculus of the ideas immanent in nervous activity*. In Bulletin of Mathematical Biophysics, 1943. (Cited on page 3.)

[Meyer & Kuschk 2019a] M. Meyer and G. Kuschk. *Deep Learning Based 3D Object Detection for Automotive Radar and Camera*. In IEEE European Radar Conference (EuRAD), 2019. (Cited on page 52.)

[Meyer & Kuschk 2019b] Michael Meyer and Georg Kuschk. *Automotive Radar Dataset for Deep Learning Based 3D Object Detection*. In IEEE European Radar Conference (EuRAD), 2019. (Cited on page 45.)

[Meyer *et al.* 2021] Michael Meyer, Georg Kuschk and Sven Tomforde. *Graph Convolutional Networks for 3D Object Detection on Radar Data*. In IEEE International Conference on Computer Vision Workshop (ICCVW), 2021. (Cited on page 48.)

[Milletari *et al.* 2016] Fausto Milletari, Nassir Navab and Seyed-Ahmad Ahmadi. *V-Net: Fully Convolutional Neural Networks for Volumetric Medical Image Segmentation*. In IEEE International Conference on 3D Vision (3DV), 2016. (Cited on page 90.)

[Minsky & Papert 1969] Marvin Minsky and Seymour Papert. Perceptrons: An Introduction to Computational Geometry. MIT Press, 1969. (Cited on pages 3 and 17.)

[Mohta *et al.* 2020] Abhishek Mohta, Fang-Chieh Chou, Brian C. Becker, Carlos Vallespi-Gonzalez and Nemanja Djuric. *Investigating the Effect of Sensor Modalities in Multi-Sensor Detection-Prediction Models*. In Conference on Neural Information Processing Systems (NeurIPS), 2020. (Cited on page 55.)

[Molchanov *et al.* 2015] Pavlo Molchanov, Shalini Gupta, Kihwan Kim and Kari Pulli. *Multi-sensor system for driver's hand-gesture recognition*. In IEEE International Conference and Workshops on Automatic Face and Gesture Recognition (FG), 2015. (Cited on page 42.)

[Molchanov 2014] P. Molchanov. *Radar Target Classification by Micro-Doppler Contributions*. PhD thesis, Tampere University, Finland, 2014. (Cited on page 11.)

[Mostajabi *et al.* 2020] Mohammadreza Mostajabi, Ching Ming Wang, Darsh Ranjan and Gilbert Hsyu. *High-Resolution Radar Dataset for Semi-Supervised Learning of Dynamic Objects*. In IEEE Conference on Computer Vision and Pattern Recognition Workshop (CVPRW), 2020. (Cited on page 46.)

[Mottaghi *et al.* 2014] Roozbeh Mottaghi, Xianjie Chen, Xiaobai Liu, Nam-Gyu Cho, Seong-Whan Lee, Sanja Fidler, Raquel Urtasun and Alan Yuille. *The Role of Context for Object Detection and Semantic Segmentation in the Wild*. In IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2014. (Cited on page 34.)

[Nabati & Qi 2020] Ramin Nabati and Hairong Qi. *Radar-Camera Sensor Fusion for Joint Object Detection and Distance Estimation in Autonomous Vehicles*. In IEEE International Conference on Intelligent Robots and Systems (IROS), 2020. (Cited on page 52.)

[Nabati & Qi 2021] Ramin Nabati and Hairong Qi. *CenterFusion: Center-based Radar and Camera Fusion for 3D Object Detection*. In IEEE Workshop on Applications of Computer Vision (WACV), 2021. (Cited on page 53.)

[Ng *et al.* 2020] Weichong Ng, Guohua Wang, Siddhartha, Zhiping Lin and Bhaskar Jyoti Dutta. *Range-Doppler Detection in Automotive Radar with Deep Learning*. In IEEE International Joint Conference on Neural Networks (IJCNN), 2020. (Cited on page 48.)

[Niesen & Unnikrishnan 2020] Urs Niesen and Jayakrishnan Unnikrishnan. *Camera-Radar Fusion for 3-D Depth Reconstruction*. In IEEE Intelligent Vehicles Symposium (IV), 2020. (Cited on page 52.)

[Nobis *et al.* 2019] Felix Nobis, Maximilian Geisslinger, Markus Weber, Johannes Betz and Markus Lienkamp. *A Deep Learning-based Radar and Camera Sensor Fusion Architecture for Object Detection*. In IEEE Sensor Data Fusion: Trends, Solutions, Applications (SDF), 2019. (Cited on page 52.)

[Nobis *et al.* 2021a] Felix Nobis, Felix Fent, Johannes Betz and Markus Lienkamp. *Kernel Point Convolution LSTM Networks for Radar Point Cloud Segmentation*. In Applied Sciences, 2021. (Cited on page 51.)

[Nobis *et al.* 2021b] Felix Nobis, Ehsan Shafiei, Phillip Karle, Johannes Betz and Markus Lienkamp. *Radar Voxel Fusion for 3D Object Detection*. In Applied Sciences, 2021. (Cited on page 55.)

[Noh *et al.* 2015] Hyeonwoo Noh, Seunghoon Hong and Bohyung Han. *Learning Deconvolution Network for Semantic Segmentation*. In IEEE International Conference on Computer Vision (ICCV), 2015. (Cited on page 159.)

[Nowruzi *et al.* 2020] F. E. Nowruzi, D. Kolhatkar, Prince Kapoor, E. J. Heravi, R. Laganiere, Julien Rebut and Waqas Malik. *Deep open space segmentation using automotive radar*. In IEEE International Conference on Microwaves for Intelligent Mobility (ICMIM), 2020. (Cited on pages 50 and 120.)

[Ouaknine *et al.* 2020] A. Ouaknine, A. Newson, J. Rebut, F. Tupin and P. Pérez. *CARRADA Dataset: Camera and Automotive Radar with Range-Angle-Doppler Annotations*. In IEEE International Conference on Pattern Recognition (ICPR), 2020. (Cited on pages 14, 58, 71 and 162.)

[Ouaknine *et al.* 2021] Arthur Ouaknine, Alasdair Newson, Patrick Pérez, Florence Tupin and Julien Rebut. *Multi-View Radar Semantic Segmentation*. In IEEE International Conference on Computer Vision (ICCV), 2021. (Cited on page 85.)

[Palffy *et al.* 2020] Andras Palffy, Jiaao Dong, Julian F. P. Kooij and Dariu M. Gavrila. *CNN based Road User Detection using the 3D Radar Cube*. In IEEE Robotics and Automation Letter (RA-L), 2020. (Cited on page 47.)

[Pegoraro & Rossi 2021] Jacopo Pegoraro and Michele Rossi. *Real-time People Tracking and Identification from Sparse mm-Wave Radar Point-clouds*. In IEEE Access, 2021. (Cited on page 46.)

[Pham & Lefevre 2021] Minh-Tan Pham and Sebastien Lefevre. *Very high resolution Airborne PolSAR Image Classification using Convolutional Neural Networks*. In IEEE European Conference on Synthetic Aperture (EUSAR), 2021. (Cited on page 43.)

[Pinheiro *et al.* 2015] Pedro O. Pinheiro, Ronan Collobert and Piotr Dollár. *Learning to Segment Object Candidates*. In Conference on Neural Information Processing Systems (NeurIPS), 2015. (Cited on page 36.)

[Pinheiro *et al.* 2016] Pedro O. Pinheiro, Tsung-Yi Lin, Ronan Collobert and Piotr Dollàr. *Learning to Refine Object Segments*. In European Conference on Computer Vision (ECCV), 2016. (Cited on page 36.)

[Prophet *et al.* 2019] Robert Prophet, Gang Li, Christian Sturm and Martin Vossiek. *Semantic Segmentation on Automotive Radar Maps*. In IEEE Intelligent Vehicles Symposium (IV), 2019. (Cited on page 50.)

[Prophet *et al.* 2020] Robert Prophet, Anastasios Deligiannis, Juan-Carlos Fuentes-Michel, Ingo Weber and Martin Vossiek. *Semantic Segmentation on 3D Occupancy Grids for Automotive Radar*. In IEEE Access, 2020. (Cited on page 50.)

[Pérez *et al.* 2019] Rodrigo Pérez, Falk Schubert, Ralph Rasshofer and Erwin Biebl. *Deep Learning Radar Object Detection and Classification for Urban Automotive Scenarios*. In Kleinheubach Conference, 2019. (Cited on page 48.)

[Qi *et al.* 2017a] Charles R. Qi, Hao Su, Mo Kaichun and Leonidas J. Guibas. *PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation*. In IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017. (Cited on pages 5, 39, 46, 109 and 128.)

[Qi *et al.* 2017b] Charles R. Qi, Li Yi, Hao Su and Leonidas J. Guibas. *PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space*. In Conference on Neural Information Processing Systems (NeurIPS), 2017. (Cited on pages 39, 48, 50, 109 and 128.)

[Qi *et al.* 2018] Charles R. Qi, Wei Liu, Chenxia Wu, Hao Su and Leonidas J. Guibas. *Frustum PointNets for 3D Object Detection from RGB-D Data*. In IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2018. (Cited on pages 48 and 55.)

[Qian *et al.* 2021] Kun Qian, Shilin Zhu, Xinyu Zhang and Li Erran Li. *Robust Multimodal Vehicle Detection in Foggy Weather Using Complementary Lidar and Radar Signals*. In IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2021. (Cited on page 54.)

[Rahnemoonfar *et al.* 2020] Maryam Rahnemoonfar, Masoud Yari and John Paden. *Radar Sensor Simulation with Generative Adversarial Network*. In IEEE International Geoscience and Remote Sensing Symposium (IGARSS), 2020. (Cited on page 43.)

[Rebut *et al.* 2022] Julien Rebut, Arthur Ouaknine, Waqas Malik and Patrick Pérez. *Raw High-Definition Radar for Multi-Task Learning*. In IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2022. (Cited on page 111.)

[Redmon & Farhadi 2017] Joseph Redmon and Ali Farhadi. *YOLO9000: Better, Faster, Stronger*. In IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017. (Cited on pages 157 and 158.)

[Redmon & Farhadi 2018] Joseph Redmon and Ali Farhadi. *YOLOv3: An Incremental Improvement*. In ArXiv, 2018. (Cited on page 53.)

[Redmon *et al.* 2016] Joseph Redmon, Santosh Divvala, Ross Girshick and Ali Farhadi. *You Only Look Once: Unified, Real-Time Object Detection*. In IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016. (Cited on page 32.)

[Ren *et al.* 2015] Shaoqing Ren, Kaiming He, Ross Girshick and Jian Sun. *Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks*. In Conference on Neural Information Processing Systems (NeurIPS), 2015. (Cited on pages 31 and 36.)

[Robbins & Monro 1951] Herbert Robbins and Sutton Monro. *A Stochastic Approximation Method*. In The Annals of Mathematical Statistics, 1951. (Cited on page 19.)

[Rohling 1983] Hermann Rohling. *Radar CFAR Thresholding in Clutter and Multiple Target Situations*. In Transactions on Aerospace and Electronic Systems, 1983. (Cited on pages 14, 74 and 96.)

[Ronneberger *et al.* 2015] Olaf Ronneberger, Philipp Fischer and Thomas Brox. *U-Net: Convolutional Networks for Biomedical Image Segmentation*. In International Conference on Medical Image Computing and Computer Assisted Intervention (MICCAI), 2015. (Cited on pages 35, 36, 50, 67, 69, 87, 93, 94, 98, 99 and 165.)

[Rosenblatt 1958] F. Rosenblatt. *The perceptron: A probabilistic model for information storage and organization in the brain.* In Psychological Review, 1958. (Cited on pages 3 and 16.)

[Russakovsky *et al.* 2015] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg and Li Fei-Fei. *ImageNet Large Scale Visual Recognition Challenge*. In International Journal of Computer Vision (IJCV), 2015. (Cited on page 29.)

[Scheiner *et al.* 2020] Nicolas Scheiner, Florian Kraus, Fangyin Wei, Buu Phan, Fahim Mannan, Nils Appenrodt, Werner Ritter, Jürgen Dickmann, Klaus Dietmayer, Bernhard Sick and Felix Heide. *Seeing Around Street Corners: Non-Line-of-Sight Detection and Tracking In-the-Wild Using Doppler Radar*. In IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2020. (Cited on page 48.)

[Scherer *et al.* 2021] Moritz Scherer, Michele Magno, Jonas Erb, Philipp Mayer, Manuel Eggimann and Luca Benini. *TinyRadarNN: Combining Spatial and Temporal Convolutional Neural Networks for Embedded Gesture Recognition with Short Range Radars*. In IEEE Internet of Things Journal (IoT-J), 2021. (Cited on page 42.)

[Schumann *et al.* 2018] Ole Schumann, Markus Hahn, Jurgen Dickmann and Christian Wohler. *Semantic Segmentation on Radar Point Clouds*. In IEEE International Conference on Information Fusion (FUSION), 2018. (Cited on page 50.)

[Schumann *et al.* 2021] Ole Schumann, Markus Hahn, Nicolas Scheiner, Fabio Weishaupt, Julius F. Tilly, Jürgen Dickmann and Christian Wöhler. *RadarScenes: A Real-World Radar Point Cloud Data Set for Automotive Applications*. In ArXiv, 2021. (Cited on pages 46 and 51.)

[Shah *et al.* 2020] Meet Shah, Zhiling Huang, Ankit Laddha, Matthew Langford, Blake Barber, Sidney Zhang, Carlos Vallespi-Gonzalez and Raquel Urtasun. *LiRaNet: End-to-End Trajectory Prediction using Spatio-Temporal Radar Fusion*. In Conference on Robot Learning (CoRL), 2020. (Cited on pages 54 and 55.)

[Sheeny *et al.* 2020] Marcel Sheeny, Andrew Wallace and Sen Wang. *300 GHz Radar Object Recognition based on Deep Neural Networks and Transfer Learning*. In IET Radar, Sonar and Navigation, 2020. (Cited on page 47.)

[Sheeny *et al.* 2021] Marcel Sheeny, Emanuele De Pellegrin, Saptarshi Mukherjee, Alireza Ahrabian, Sen Wang and Andrew Wallace. *RADIATE: A Radar Dataset for Automotive Perception*. In IEEE International Conference on Robotics and Automation (ICRA), 2021. (Cited on page 45.)

[Shuai *et al.* 2021] Xian Shuai, Yulin Shen, Yi Tang, Shuyao Shi, Luping Ji and Guoliang Xing. *milliEye: A Lightweight mmWave Radar and Camera Fusion System for Robust Object Detection*. In IEEE International Conference on Internet of Things Design and Implementation (IoTDI), 2021. (Cited on page 53.)

[Silver *et al.* 2016] David Silver, Aja Huang, Chris J. Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel and Demis Hassabis. *Mastering the game of Go with deep neural networks and tree search*. In Nature, 2016. (Cited on page 4.)

[Simonyan & Zisserman 2015] Karen Simonyan and Andrew Zisserman. *Very Deep Convolutional Networks for Large-Scale Image Recognition*. In International Conference on Learning Representations (ICLR), 2015. (Cited on pages 27, 66 and 93.)

[Sless *et al.* 2019] Liat Sless, Gilad Cohen, Bat El Shlomo and Shaul Oron. *Road Scene Understanding by Occupancy Grid Learning from Sparse Radar Clusters using Semantic Segmentation*. In IEEE International Conference on Computer Vision Workshop (ICCVW), 2019. (Cited on page 50.)

[Srivastava *et al.* 2014] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever and Ruslan Salakhutdinov. *Dropout: A Simple Way to Prevent Neural Networks from Overfitting*. In Journal of Machine Learning Research, 2014. (Cited on page 21.)

[Stephan *et al.* 2021] Michael Stephan, Avik Santra and Georg Fischer. *Human Target Detection and Localization with Radars Using Deep Learning*. In Deep Learning Applications, 2021. (Cited on page 48.)

[Stroescu *et al.* 2020] Ana Stroescu, Liam Daniel, Dominic Phippen, Mikhail Cherniakov and Marina Gashinova. *Object Detection on Radar Imagery for Autonomous Driving Using Deep Neural Networks*. In IEEE European Radar Conference (EuRAD), 2020. (Cited on page 47.)

[Sun *et al.* 2017] Chen Sun, Abhinav Shrivastava, Saurabh Singh and Abhinav Gupta. *Revisiting Unreasonable Effectiveness of Data in Deep Learning Era*. In IEEE International Conference on Computer Vision (ICCV), 2017. (Cited on pages 37 and 38.)

[Sun *et al.* 2019] Yuliang Sun, Tai Fei, Shangyin Gao and Nils Pohl. *Automatic Radar-based Gesture Detection and Classification via a Region-based Deep Convolutional Neural Network*. In IEEE International Conference on Acoustics, Speech, & Signal Processing (ICASSP), 2019. (Cited on page 42.)

[Sun *et al.* 2020] Pei Sun, Henrik Kretzschmar, Xerxes Dotiwalla, Aurelien Chouard, Vijaysai Patnaik, Paul Tsui, James Guo, Yin Zhou, Yuning Chai, Benjamin Caine, Vijay Vasudevan, Wei Han, Jiquan Ngiam, Hang Zhao, Aleksei Timofeev, Scott Ettinger, Maxim Krivokon, Amy Gao, Aditya Joshi, Yu Zhang, Jonathon Shlens, Zhifeng Chen and Dragomir Anguelov. *Scalability in Perception for Autonomous Driving: Waymo Open Dataset*. In IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2020. (Cited on page 5.)

[Sun *et al.* 2021] Pengliang Sun, Xuetong Niu, Pengfei Sun and Kele Xu. *Squeeze-and-Excitation network-Based Radar Object Detection With Weighted Location Fusion*. In ACM International Conference on Multimedia Retrieval (ICMR), 2021. (Cited on page 47.)

[Svenningsson *et al.* 2021] Peter Svenningsson, Francesco Fioranelli and Alexander Yarovoy. *Radar-PointGNN: Graph Based Object Recognition for Unstructured Radar Point-cloud Data*. In IEEE Radar Conference (RadarConf), 2021. (Cited on page 48.)

[Szegedy *et al.* 2015] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke and Andrew Rabinovich. *Going deeper with convolutions*. In IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2015. (Cited on pages 28 and 32.)

[Szegedy *et al.* 2016] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens and Z. Wojna. *Rethinking the Inception Architecture for Computer Vision*. In IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016. (Cited on page 28.)

[Szegedy *et al.* 2017] Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke and Alexander A. Alemi. *Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning*. In AAAI Conference on Artificial Intelligence, 2017. (Cited on page 28.)

[Taigman *et al.* 2014] Yaniv Taigman, Ming Yang, Marc'Aurelio Ranzato and Lior Wolf. *DeepFace: Closing the Gap to Human-Level Performance in Face Verification*. In IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2014. (Cited on page 4.)

[Touvron *et al.* 2021] Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles and Hervé Jégou. *Training data-efficient image transformers & distillation through attention*. In International Conference on Machine Learning (ICML), 2021. (Cited on page 29.)

[Truong & Yanushkevich 2019] Thomas Truong and Svetlana Yanushkevich. *Generative Adversarial Network for Radar Signal Synthesis*. In IEEE International Joint Conference on Neural Networks (IJCNN), 2019. (Cited on page 43.)

[Uijlings *et al.* 2013] J. R. R. Uijlings, K. E. A. van de Sande, T. Gevers and A.W.M. Smeulders. *Selective Search for Object Recognition*. In International Journal of Computer Vision (IJCV), 2013. (Cited on page 30.)

[Wallace *et al.* 2021] Andrew M. Wallace, Saptarshi Mukherjee, Bemsibom Toh and Alireza Ahrabian. *Combining automotive radar and LiDAR for surface detection in adverse conditions*. In IET Radar, Sonar & Navigation, 2021. (Cited on page 54.)

[Wang *et al.* 2016] Saiwen Wang, Jie Song, Jaime Lien, Ivan Poupyrev and Otmar Hilliges. *Interacting with Soli: Exploring Fine-Grained Dynamic Gesture Recognition in the Radio-Frequency Spectrum*. In ACM User Interface Software and Technology (UIST), 2016. (Cited on page 42.)

[Wang *et al.* 2019] Yong Wang, Xiuqian Jia, Mu Zhou, Xiaolong Yang and Zengshan Tian. *Rammar: RAM Assisted Mask R-CNN for FMCW Sensor Based HGD System*. In IEEE International Conference on Communications (ICC), 2019. (Cited on page 42.)

[Wang *et al.* 2020a] Leichen Wang, Tianbai Chen, Carsten Anklam and Bastian Goldluecke. *High Dimensional Frustum PointNet for 3D Object Detection from Camera, LiDAR, and Radar*. In IEEE Intelligent Vehicles Symposium (IV), 2020. (Cited on page 55.)

[Wang *et al.* 2020b] Yuchen Wang, Mingze Xu, John Paden, Lora Koenig, Geoffrey Fox and David Crandall. *Deep Tiered Image Segmentation forDetecting Internal Ice*

Layers in Radar Imagery. In IEEE International Conference on Multimedia & Expo (ICME), 2020. (Cited on page 43.)

[Wang *et al.* 2021a] Yizhou Wang, Zhongyu Jiang, Yudong Li, Jenq-Neng Hwang, Guanbin Xing and Hui Liu. *RODNet: A Real-Time Radar Object Detection Network Cross-Supervised by Camera-Radar Fused Object 3D Localization*. In Journal of Selected Topics in Signal Processing, 2021. (Cited on page 47.)

[Wang *et al.* 2021b] Yizhou Wang, Gaoang Wang, Hung-Min Hsu, Hui Liu and Jenq-Neng Hwang. *Rethinking of Radar's Role: A Camera-Radar Dataset and Systematic Annotator via Coordinate Alignment*. In IEEE Conference on Computer Vision and Pattern Recognition Workshop (CVPRW), 2021. (Cited on pages 44 and 47.)

[Wang *et al.* 2021c] Zeyu Wang, Chenglin Yao, Jianfeng Ren and Xudong Jiang. *Human Activity Recognition Using 3D Orthogonally-projected EfficientNet on Radar Time-Range-Doppler Signature*. In ArXiv, 2021. (Cited on page 42.)

[Weston *et al.* 2019] Rob Weston, Sarah Cen, Paul Newman and Ingmar Posner. *Probably Unknown: Deep Inverse Sensor Modelling Radar*. In IEEE International Conference on Robotics and Automation (ICRA), 2019. (Cited on page 54.)

[Weston *et al.* 2021] Rob Weston, Oiwi Parker Jones and Ingmar Posner. *There and Back Again: Learning to Simulate Radar Data for Real-World Applications*. In IEEE International Conference on Robotics and Automation (ICRA), 2021. (Cited on page 54.)

[Xie *et al.* 2017] Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu and Kaiming He. *Aggregated Residual Transformations for Deep Neural Networks*. In IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017. (Cited on pages 33, 36 and 160.)

[Xu *et al.* 2021] Baowei Xu, Xinyu Zhang, Li Wang, Xiaomei Hu, Zhiwei Li, Shuyue Pan, Jun Li and Yongqiang Deng. *RPFA-Net: a 4D RaDAR Pillar Feature Attention Network for 3D Object Detection*. In IEEE International Conference on Intelligent Transportation Systems (ITSC), 2021. (Cited on page 49.)

[Yadav *et al.* 2020] Ritu Yadav, Axel Vierling and Karsten Berns. *Radar+RGB Attentive Fusion for Robust Object Detection in Autonomous Vehicles*. In IEEE International Conference on Image Processing (ICIP), 2020. (Cited on page 52.)

[Yang *et al.* 2018] Bin Yang, Wenjie Luo and Raquel Urtasun. *PIXOR: Real-time 3D Object Detection from Point Clouds*. In IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2018. (Cited on pages 40, 117, 118, 119 and 120.)

[Yang *et al.* 2020] Bin Yang, Runsheng Guo, Ming Liang, Sergio Casas and Raquel Urtasun. *RadarNet: Exploiting Radar for Robust Perception of Dynamic Objects*. In European Conference on Computer Vision (ECCV), 2020. (Cited on page 54.)

[Yu & Koltun 2016] Fisher Yu and Vladlen Koltun. *Multi-Scale Context Aggregation by Dilated Convolutions*. In International Conference on Learning Representations (ICLR), 2016. (Cited on page 159.)

[Yu *et al.* 2020] Fisher Yu, Wenqi Xian, Yingying Chen, Fangchen Liu, Mike Liao, Vashisht Madhavan and Trevor Darrell. *BDD100K: A Diverse Driving Video Database with Scalable Annotation Tooling*. In IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2020. (Cited on page 5.)

[Zeiler & Fergus 2014] Matthew D. Zeiler and Rob Fergus. *Visualizing and Understanding Convolutional Networks*. In European Conference on Computer Vision (ECCV), 2014. (Cited on page 31.)

[Zhang *et al.* 2017a] Hang Zhang, Jia Xue and Kristin Dana. *Deep TEN: Texture Encoding Network*. In IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017. (Cited on page 160.)

[Zhang *et al.* 2017b] Zhimian Zhang, Haipeng Wang, Feng Xu and Ya-Qiu Jin. *Complex-Valued Convolutional Neural Network and Its Application in Polarimetric SAR Image Classification*. In Transactions on Geoscience and Remote Sensing, 2017. (Cited on page 43.)

[Zhang *et al.* 2018a] Hang Zhang, Kristin Dana, Jianping Shi, Zhongyue Zhang, Xiaogang Wang, Ambrish Tyagi and Amit Agrawal. *Context Encoding for Semantic Segmentation*. In IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2018. (Cited on page 160.)

[Zhang *et al.* 2018b] Zhenyuan Zhang, Zengshan Tian and Mu Zhou. *Latern: Dynamic Continuous Hand Gesture Recognition Using FMCW Radar Sensor*. In Sensors Journal, 2018. (Cited on page 42.)

[Zhang *et al.* 2019] Zhenyuan Zhang, Zengshan Tian, Ying Zhang, Mu Zhou and Bang Wang. *u-DeepHand: FMCW Radar-Based Unsupervised Hand Gesture Feature Learning Using Deep Convolutional Auto-Encoder Network*. In Sensors Journal, 2019. (Cited on page 42.)

[Zhang *et al.* 2020] Lamei Zhang, Siyu Zhang, Hongwei Dong and Da Lu. *Polsar Image Classification via Complex-Valued Multi-Scale Convolutional Neural Network*. In IEEE International Geoscience and Remote Sensing Symposium (IGARSS), 2020. (Cited on page 43.)

[Zhang *et al.* 2021a] Ao Zhang, Farzan Erlik Nowruzi and Robert Laganiere. *RADDet: Range-Azimuth-Doppler based Radar Object Detection for Dynamic Road Users*. In Computer Vision and Robotics (CVR), 2021. (Cited on pages 44, 47, 83, 96, 97 and 125.)

[Zhang *et al.* 2021b] Jingwei Zhang, Ming Zhang, Zicheng Fang, Yulong Wang, Xian Zhao and Shiliang Pu. *RVDet: Feature-level Fusion of Radar and Camera for*

Object Detection. In IEEE International Conference on Intelligent Transportation Systems (ITSC), 2021. (Cited on page 52.)

[Zhao *et al.* 2017] Hengshuang Zhao, Jianping Shi, Xiaojuan Qi, Xiaogang Wang and Jiaya Jia. *Pyramid Scene Parsing Network*. In IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017. (Cited on pages 37 and 159.)

[Zheng *et al.* 2021] Zangwei Zheng, Xiangyu Yue, Kurt Keutzer and Alberto Sangiovanni Vincentelli. *Scene-aware Learning Network for Radar Object Detection*. In ACM International Conference on Multimedia Retrieval (ICMR), 2021. (Cited on page 48.)

[Zhou & Tuzel 2018] Yin Zhou and Oncel Tuzel. *VoxelNet: End-to-End Learning for Point Cloud Based 3D Object Detection*. In IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2018. (Cited on page 55.)

[Zhu *et al.* 2017] Jun-Yan Zhu, Taesung Park, Phillip Isola and Alexei A. Efros. *Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks*. In IEEE International Conference on Computer Vision (ICCV), 2017. (Cited on pages 43 and 52.)

[Zhu *et al.* 2020] JianPing Zhu, HaiQuan Chen and WenBin Ye. *Classification of Human Activities Based on Radar Signals using 1D-CNN and LSTM*. In IEEE International Symposium on Circuits and Systems (ISCAS), 2020. (Cited on page 42.)

Appendices

APPENDIX A

Background

A.1 Deep learning

A.1.1 Object detection

This section details deep learning methods and architectures for object detection in natural images. It provides additional details to those presented in Section 2.6.2.

Region-based Fully Convolutional Network (R-FCN). Fast and Faster R-CNN requires multiple sub-networks to process the region proposals produced by the RPN. The R-FCN proposed by [Dai *et al.* 2016] uses a single network, shared on the entire image, processing the region proposals. This module is fully convolutional except the last fully connected layer used for classification. The architecture starts with a ResNet-101 [He *et al.* 2016] processing the input and generating feature maps corresponding to the number of classes. These feature maps are called position-sensitive score maps because they take into account the spatial localisation of a particular object. There are $s \times s \times (K+1)$ score maps forming a "score bank", where s is the size of the score map and K the number of classes. The idea of these maps is to create patches recognizing a single part of an object, each patch being specialized in a category. A RPN generates RoI in parallel. The RoI extracted in the feature maps are placed into bins which are compared with the score bank. A vote is performed, if enough bins are activated (regarding the corresponding score), the patch localizes and classifies the object. The best R-FCN have reached mAP scores of 83.6% on the 2007 PASCAL VOC challenge while bein trained using the 2007, 2012 PASCAL VOC datasets and the COCO dataset. On 2015 COCO challenge, they performed a mAP of 53.2% for an IoU = 0.5 and a score of 31.5% for the official mAP metric. The authors noticed that the R-FCN is 2.5 to 20 times faster than the Faster R-CNN counterpart.

YOLO extensions. In their work, [Redmon & Farhadi 2017] improved the YOLO presented in a previous paragraph in YOLOv2 and proposed a new model, YOLO9000, capable of detecting more than 9000 categories while running in almost real time (around 10 FPS).

The YOLOv2 focuses on improving the accuracy while still being a fast detector. Batch normalization is added to prevent over-fitting without using dropout. The model is also able to process higher resolution images to potentially detect smaller objects: 608×608 instead of 448×448 in the initial version. The final fully connected layer of the YOLO model predicting the bounding box coordinates has been removed to use anchor boxes in the same way as Faster R-CNN. The input image is divided into a grid of cells, each

one containing 5 anchor boxes. YOLOv2 uses $19\times19\times5$ = 1805 anchor boxes by image instead of 98 boxes for the YOLO model. YOLOv2 predicts the correction of each anchor box relative to the location of the grid cell (ranging between 0 and 1) and selects the boxes according to their confidence as the SSD model. The dimensions of the anchor boxes have been fixed using a k-means clustering on the training set of bounding boxes. It uses a ResNet architecture to stack high and low resolution feature maps to detect smaller objects. Their proposed backbone called Darknet-19 is composed of 19 convolutional layers with 3×3 and 1×1 kernels, followed by max-pooling layers to reduce the output dimension. A final 1×1 convolutional layer outputs 5 boxes per cell of the grid with 5 coordinates and a probability for each class (20 in the case of the PASCAL VOC dataset). The YOLOv2 model trained with the 2007 and 2012 PASCAL VOC datasets obtained a 78.6% mAP score on the 2007 PASCAL VOC challenge at 40 FPS during inference. The model trained with the 2015 COCO dataset got mAP scores on the corresponding challenge of 44.0% for an IoU threshold of 0.5, 19.2% for an IoU threshold of 0.75 and 21.6% for the official mAP metric.

To train their second proposed architecture, the authors have combined the ImageNet dataset with the COCO dataset. The ImageNet dataset for classification contains 1000 categories and the 2015 COCO dataset only 80 categories. The ImageNet classes are based on the WordNet[1] lexicon developed by the Princeton University which is composed of more than 20,000 words. [Redmon & Farhadi 2017] detail a method to construct a tree version of the WordNet. The model predicting on an image will use a softmax applied on a group of labels with the same hyponym[2]. The final probability associated to a label is computed with posterior probabilities in the tree. When the authors extend the concept to the entire WordNet lexicon excluding under-represented categories, they obtain more than 9,000 categories. The ImageNet and COCO dataset combination is used to train a YOLOv2 architecture with 3 prior convolution layers instead of 5 to limit the output size. This model called YOLO9000 is theoretically able to learn categories of the presented lexicon. It is evaluated on the ImageNet dataset for the detection task with around 200 labels. Only 44 labels are shared between the training and the testing dataset so the results are not significant. It gets a 19.7% mAP score overall the test dataset.

A.1.2 Segmentation

This section details deep learning methods and architectures for semantic segmentation of natural images. It provides additional details to those presented in Section 2.6.3.

ParseNet. In their work, [Liu *et al.* 2015] present improvements on the FCN method [Long *et al.* 2015]. They claimed that the FCN model looses the global context of the image in its deep layers by specializing the generated feature maps. The ParseNet is an end-to-end convolutional network with "contexture" modules. This module takes feature maps as input and processes them with two pathways. The first one aggregates the feature

[1] https://wordnet.princeton.edu/

[2] An hyponym is a word of more specific meaning than a general or superordinate term applicable to. Source: Oxford Languages.

maps with a global pooling to obtain global features normalised with an L2-norm which are finally up-pooled (replicated) to recover the original dimensions. The second pathway simply normalises the input feature maps with a L2-norm. The outputs of both pathways are stacked and transmitted to the next layer. The normalization is helpful to scale the concatenated feature maps values and it leads to better performances. The ParseNet is a FCN with contexture modules instead of simple convolutional layers. It obtained a 69.8% mIoU score on the 2012 PASCAL VOC segmentation challenge and a 40.4% mIoU score on the PASCAL-Context challenge.

Convolution and deconvolutional networks. [Noh *et al.* 2015] proposed an end-to-end architecture composed of two linked parts. The first one is a convolutional network with a VGG16 architecture with only two fully connected layers. It takes as input a proposal, *e.g.* a bounding box generated by an object detection model. The proposal is processed and transformed by the convolutional network to generate a vector of features. The deconvolutional network takes the vector of features as input and aims to classify pixel-wise the entire image. The deconvolutional network is composed of successive layers of un-pooling and deconvolutions. The un-pooling targets the location of the maximum value in the mirrored feature map reduced by the max-pooling in the first convolutional network. It expands the feature map with sparse values, the value taken as input is placed at the recorded position while the others are filled with zeros. The deconvolution takes a single value as input and generates a $k \times k$ output depending of the kernel size. It aims in learning the expansion of the feature maps while recovering the dimension of the input and keeping the information dense. The authors analyzed deconvolution feature maps and noticed that low-level ones are specific to the shape while the higher-level ones help to classify the proposal. Finally, when all the proposals of an image are processed by the entire network, the maps are concatenated to obtain the fully segmented image. This network obtained a 72.5% mIoU on the 2012 PASCAL VOC segmentation challenge.

Pyramid scene parsing network (PSPNet). [Zhao *et al.* 2017] developed the PSPNet to better learn the global context representation of a scene. Patterns are extracted from the input image using a ResNet [He *et al.* 2016] as feature extractor with a dilated network strategy[3]. The feature maps are used as input of a pyramid pooling module to distinguish patterns with different scales. This pooling layer consists in four parallel convolutions with different kernel sizes. Their respective outputs are processed by 1×1 convolutions to reduce the number of feature maps and up-sampled using bilinear interpolation to match their dimensions. Each pyramid level analyses sub-regions of the image at different locations. The outputs of the pyramid levels are concatenated to the initial feature maps. The output thus contains the local and the global context information. They are finally processed by a convolutional layer to generate the pixel-wise predictions. The best PSPNet with a pre-trained ResNet (using the COCO dataset) reached a 85.4% mIoU score on the 2012 PASCAL VOC semantic segmentation challenge.

[3]The dilated convolutional layer has been proposed by [Yu & Koltun 2016]. It consists in a convolutional layer with an expanded kernel (the neurons of the kernel are no more side-by-side). A dilation rate fixes the gap between two neurons in term of pixel. More details are provided in the DeepLab paragraph.

Path aggregation network (PANet). In their work, [Liu *et al.* 2018] proposed the PANet, an improvement of the information propagation in the Mask R-CNN and FPN frameworks. The feature extractor of the network uses a FPN architecture with an additional augmented bottom-up pathway of convolutions improving the propagation of low-layer features. Each stage of this third pathway takes as input the feature maps of the previous stage and processes them with a 3×3 convolutional layer. The output is summed with the same stage feature maps of the top-down pathway using lateral connections. These maps are used as input to the next stage.

The feature maps of the augmented bottom-up pathway are pooled with a RoIAlign layer to extract proposals from all level features. An adaptive feature pooling layer processes the maps of each stage with a fully connected layer and concatenate all the outputs. The output of the adaptive feature pooling layer is taken as input of three branches similarly to the Mask R-CNN. The two first branches uses a fully connected layer to generate the predictions of the regression for the location and the classification. The third branch processes the RoI with a FCN to predict a binary pixel-wise mask for the detected object. An additional path processes the output of a convolutional layer of the FCN with a fully connected layer improving the location of the predicted pixels. Finally the output of the parallel path is reshaped and concatenated to the output of the FCN generating the binary mask.

The PANet achieved 42.0% AP score on the 2016 COCO segmentation challenge using a ResNeXt as feature extractor. They also performed the 2017 COCO segmentation challenge with an 46.7% AP score using a ensemble of seven feature extractors: ResNet [He *et al.* 2016], ResNeXt [Xie *et al.* 2017] and SENet [Hu *et al.* 2018].

Context encoding network (EncNet). [Zhang *et al.* 2018a] created the EncNet architecture capturing global information in an image to improve semantic segmentation. The model starts by using a ResNet [He *et al.* 2016] backbone followed by a "context encoding" module inspired from the encoding layer of [Zhang *et al.* 2017a]. It learns visual centers and smoothing factors creating an embedding of contextual information while specializing feature maps by class. The final layer of the module learns scaling factors of contextual information by class with an attention layer (fully connected layer). Theses factors are applied to the output feature maps of the backbone network via a skip connection. In parallel, an additional fully connected layer classifies the global context of the module with a softmax activation. This branch is trained with a Binary Cross-Entropy (BCE) loss, named semantic encoding loss (SE-Loss), regularizing the training of the module with the entire context (at the contrary to pixel-wise loss). The outputs of the context encoding module are reshaped and processed by a dilated convolution while minimizing two SE-losses and a final pixel-wise loss. The best EncNet reached 52.6% mIoU on the PASCAL-Context challenge. It also achieved a 85.9% mIoU score on the 2012 PASCAL VOC segmentation challenge.

APPENDIX B

RADAR scene understanding

B.1 Multi-view RADAR semantic segmentation

B.1.1 RAD tensor visualisation

An illustration of the RAD tensor is proposed in Figure B.1. Each slice of 2D views corresponds to a discretized bin of the third axis. In Figure B.1(b) for instance, the 256 range-Doppler slices correspond to the view of each discretized value of the angle axis. One can observe redundant signal information and a significant level of noise for each group of 2D-view slices.

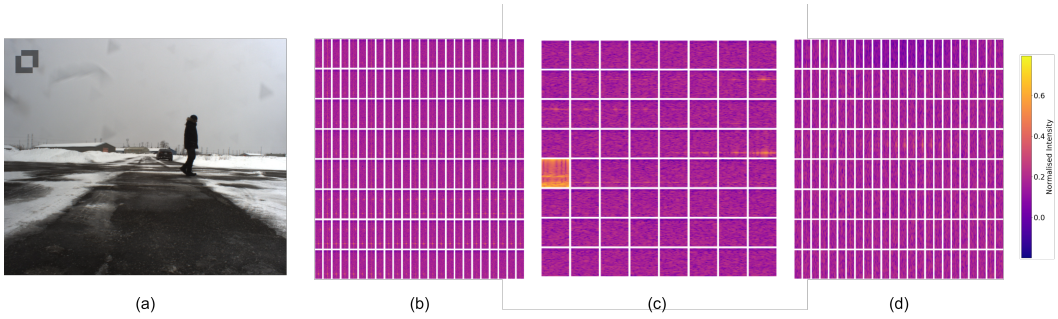


Figure B.1: **Visualisation of the Range-Angle-Doppler (RAD) tensor**. (a) Camera image of the scene. The corresponding RAD tensor is visualised by slices of (b) range-Doppler, (c) range-angle or (d) angle-Doppler views w.r.t. their discretized third axis.

B.1.2 Detailed multi-view architectures

The architecture details of the proposed multi-view network (MV-Net) and temporal multi-view network with ASPP modules (TMVA-Net) are respectively provided in Tables B.1 and B.2. For each layer, the parameters of the operations used are specified in the following manner:

- n-dim convolution: Conv$n$D (input_channels, output_channels, kernel_size, stride, padding, dilation_rate);
- n-dim up-convolution: ConvTranspose$n$D (input_channels, output_channels, kernel_size, stride, padding, dilation_rate);
- maximum pooling: MaxPool2D (kernel_size, stride);

- atrous spatial pyramid pooling: ASPP (input_channels, output_channels);
- n-D batch normalisation: BNnD (input_channels);
- Leaky ReLU activation: LeakyReLU (negative_slope);

Where $n \in \{1, 2, 3\}$ is the dimension of the associated operation.

The ASPP module [Chen *et al.* 2018a] has the same architecture as the one introduced by Kaul *et al.* [Kaul *et al.* 2020] for range-angle semantic segmentation. We note that the 'output_channels' parameter for the ASPP module corresponds to the number of output channels for each parallel convolution. We also note that the 'stride' parameter can be either a scalar or a tuple of scalars depending on the axis on which it is applied.

B.1.3 Pre-processing and training procedures

The experiments in the main paper have been conducted using the parameters detailed in Table B.3. An exponential decay with $\gamma = 0.9$ has been applied to each learning rate with an epoch step specific to each model (see Table B.3). The competing methods have been trained using the Cross Entropy (CE) loss, except for the RSS-Net, which is trained with a weighted Cross Entropy (wCE) using the formulation in [Kaul *et al.* 2020]. Our methods have been trained with the proposed combination of losses using the following parameters set up empirically: $\lambda_{\text{wCE}} = 1$, $\lambda_{\text{SDice}} = 10$ and $\lambda_{\text{CoL}} = 5$.

The architectures with which we compare our work have been designed to process inputs of size 256×256. Since the size of the range-Doppler view is 256×64 in the CARRADA dataset [Ouaknine *et al.* 2020], it is resized in the Doppler dimension to train these competing models. On the other hand, the proposed architectures are composed of down-sampling layers adapted to the size of the Doppler dimension, thus they do not require this pre-processing step. The range-angle view has a size of 256×256 and does not require a resizing in both cases. For all methods, we used vertical and horizontal flip as data augmentation to reduce over-fitting.

Each view is normalised between 0 and 1 using local batch statistics for the competing methods. Our normalisation strategy consists in using the global statistics of the entire CARRADA dataset to normalise the input views.

B.1.4 Quantitative results

The proposed TMVA-Net architecture provides the best trade-off between performance and number of parameters for both range-Doppler and range-angle semantic segmentation tasks, as illustrated in Figure B.2 with mIoU metric.

B.1.5 Variability of the quantitative results by method

This section details a study of performance variability of the proposed and competing methods. Each model has been trained four times using the CARRADA-Train and CARRADA-Val datasets following the procedures explained in Section 5.1.3. Three of the four models have been trained with different weight initialisation. The training process of the fourth

	Layer	Inputs	Output resolution $(C \times H \times W)$	Operations
RD Encoder	rd_layer1	RD view	$128 \times 256 \times 64$	Conv2D(3, 128, 3×3, 1, 1, 1) + BN2D(128) + LeakyReLU(0.01) Conv2D(128, 128, 3×3, 1, 1, 1) + BN2D(128) + LeakyReLU(0.01)
	rd_layer2	rd_layer1	$128 \times 128 \times 64$	MaxPool2D(2, (2, 1))
	rd_layer3	rd_layer2	$128 \times 128 \times 64$	Conv2D(128, 128, 3×3, 1, 1, 1) + BN2D(128) + LeakyReLU(0.01) Conv2D(128, 128, 3×3, 1, 1, 1) + BN2D(128) + LeakyReLU(0.01)
	rd_layer4	rd_layer3	$128 \times 64 \times 64$	MaxPool2D(2, (2, 1))
	rd_layer5	rd_layer4	$128 \times 64 \times 64$	Conv1D(128, 128, 1×1, 1, 0, 1)
RA Encoder	ra_layer1	RA view	$128 \times 256 \times 256$	Conv2D(3, 128, 3×3, 1, 1, 1) + BN2D(128) + LeakyReLU(0.01) Conv2D(128, 128, 3×3, 1, 1, 1) + BN2D(128) + LeakyReLU(0.01)
	ra_layer2	ra_layer1	$128 \times 128 \times 128$	MaxPool2D(2, 2)
	ra_layer3	ra_layer2	$128 \times 128 \times 128$	Conv2D(128, 128, 3×3, 1, 1, 1) + BN2D(128) + LeakyReLU(0.01) Conv2D(128, 128, 3×3, 1, 1, 1) + BN2D(128) + LeakyReLU(0.01)
	ra_layer4	ra_layer3	$128 \times 64 \times 64$	MaxPool2D(2, 2)
	ra_layer5	ra_layer4	$128 \times 64 \times 64$	Conv1D(128, 128, 1×1, 1, 0, 1)
Latent space	layer6	rd_layer5, ra_layer5	$256 \times 64 \times 64$	concatenate(rd_layer5, ra_layer5)
RD Decoder	rd_layer7	layer6	$128 \times 64 \times 64$	Conv1D(256, 128, 1×1, 1, 0, 1)
	rd_layer8	rd_layer7	$128 \times 128 \times 64$	ConvTranspose2D(128, 128, 2×1, (2, 1), 1, 1)
	rd_layer9	rd_layer8	$128 \times 128 \times 64$	Conv2D(128, 128, 3×3, 1, 1, 1) + BN2D(128) + LeakyReLU(0.01) Conv2D(128, 128, 3×3, 1, 1, 1) + BN2D(128) + LeakyReLU(0.01)
	rd_layer10	rd_layer9	$128 \times 256 \times 64$	ConvTranspose2D(128, 128, 2×1, (2, 1), 1, 1)
	rd_layer11	rd_layer10	$128 \times 256 \times 64$	Conv2D(128, 128, 3×3, 1, 1, 1) + BN2D(128) + LeakyReLU(0.01) Conv2D(128, 128, 3×3, 1, 1, 1) + BN2D(128) + LeakyReLU(0.01)
	rd_layer12	rd_layer11	$K \times 256 \times 64$	Conv1D(128, K, 1×1, 1, 0, 1)
RA Decoder	ra_layer7	layer6	$128 \times 64 \times 64$	Conv1D(256, 128, 1×1, 1, 0, 1)
	ra_layer8	ra_layer7	$128 \times 128 \times 128$	ConvTranspose2D(128, 128, 2×2, 2, 1, 1)
	ra_layer9	ra_layer8	$128 \times 128 \times 128$	Conv2D(128, 128, 3×3, 1, 1, 1) + BN2D(128) + LeakyReLU(0.01) Conv2D(128, 128, 3×3, 1, 1, 1) + BN2D(128) + LeakyReLU(0.01)
	ra_layer10	ra_layer9	$128 \times 256 \times 256$	ConvTranspose2D(128, 128, 2×2, 2, 1, 1)
	ra_layer11	ra_layer10	$128 \times 256 \times 256$	Conv2D(128, 128, 3×3, 1, 1, 1) + BN2D(128) + LeakyReLU(0.01) Conv2D(128, 128, 3×3, 1, 1, 1) + BN2D(128) + LeakyReLU(0.01)
	ra_layer12	ra_layer11	$K \times 256 \times 256$	Conv1D(128, K, 1×1, 1, 0, 1)

Table B.1: **Multi-view network (MV-Net) architecture**. This table lists all the layers contained in the model taking as input multi-view RADAR representations (RD and RA views) to predict segmentation maps for each multi-view output. Details about the parameters of each operation are provided in Section B.1.2. Let K be the number of classes. The number of input channels in the first layer corresponds to the consecutive frames of each view stacked in temporal dimension, here $q = 2$ and thus the number of channels is 3.

	Layer	Inputs	Output resolution ($C \times H \times W$)	Operations
RD Encoder	rd_layer1	RD view	$128 \times 256 \times 64$	Conv3D(1, 128, $3 \times 3 \times 3$, 1, (0, 1, 1), 1) + BN3D(128) + LeakyReLU(0.01) Conv3D(128, 128, $3 \times 3 \times 3$, 1, (0, 1, 1), 1) + BN3D(128) + LeakyReLU(0.01)
	rd_layer2	rd_layer1	$128 \times 128 \times 64$	MaxPool2D(2, (2, 1))
	rd_layer3	rd_layer2	$128 \times 128 \times 64$	Conv2D(128, 128, 3×3, 1, 1, 1) + BN2D(128) + LeakyReLU(0.01) Conv2D(128, 128, 3×3, 1, 1, 1) + BN2D(128) + LeakyReLU(0.01)
	rd_layer4	rd_layer3	$128 \times 64 \times 64$	MaxPool2D(2, (2, 1))
	rd_layer5	rd_layer4	$128 \times 64 \times 64$	Conv1D(128, 128, 1×1, 1, 0, 1)
	rd_layer6	rd_layer5	$640 \times 64 \times 64$	ASPP(128, 128)
	rd_layer7	rd_layer6	$128 \times 64 \times 64$	Conv1D(640, 128, 1×1, 1, 0, 1)
AD Encoder	ad_layer1	AD view	$128 \times 256 \times 64$	Conv3D(1, 128, $3 \times 3 \times 3$, 1, (0, 1, 1), 1) + BN3D(128) + LeakyReLU(0.01) Conv3D(128, 128, $3 \times 3 \times 3$, 1, (0, 1, 1), 1) + BN3D(128) + LeakyReLU(0.01)
	ad_layer2	ad_layer1	$128 \times 128 \times 64$	MaxPool2D(2, (2, 1))
	ad_layer3	ad_layer2	$128 \times 128 \times 64$	Conv2D(128, 128, 3×3, 1, 1, 1) + BN2D(128) + LeakyReLU(0.01) Conv2D(128, 128, 3×3, 1, 1, 1) + BN2D(128) + LeakyReLU(0.01)
	ad_layer4	ad_layer3	$128 \times 64 \times 64$	MaxPool2D(2, (2, 1))
	ad_layer5	ad_layer4	$128 \times 64 \times 64$	Conv1D(128, 128, 1×1, 1, 0, 1)
	ad_layer6	ad_layer5	$640 \times 64 \times 64$	ASPP(128, 128)
	ad_layer7	ad_layer6	$128 \times 64 \times 64$	Conv1D(640, 128, 1×1, 1, 0, 1)
RA Encoder	ra_layer1	RA view	$128 \times 256 \times 256$	Conv3D(1, 128, $3 \times 3 \times 3$, 1, (0, 1, 1), 1) + BN3D(128) + LeakyReLU(0.01) Conv3D(128, 128, $3 \times 3 \times 3$, 1, (0, 1, 1), 1) + BN3D(128) + LeakyReLU(0.01)
	ra_layer2	ra_layer1	$128 \times 128 \times 128$	MaxPool2D(2, 2)
	ra_layer3	ra_layer2	$128 \times 128 \times 128$	Conv2D(128, 128, 3×3, 1, 1, 1) + BN2D(128) + LeakyReLU(0.01) Conv2D(128, 128, 3×3, 1, 1, 1) + BN2D(128) + LeakyReLU(0.01)
	ra_layer4	ra_layer3	$128 \times 64 \times 64$	MaxPool2D(2, 2)
	ra_layer5	ra_layer4	$128 \times 64 \times 64$	Conv1D(128, 128, 1×1, 1, 0, 1)
	ra_layer6	ra_layer5	$640 \times 64 \times 64$	ASPP(128, 128)
	ra_layer7	ra_layer6	$128 \times 64 \times 64$	Conv1D(640, 128, 1×1, 1, 0, 1)
Latent space	layer8	rd_layer5, ra_layer5, ad_layer5	$384 \times 64 \times 64$	concatenate(rd_layer5, ra_layer5, ad_layer5)
RD Decoder	rd_layer9	layer8	$128 \times 64 \times 64$	Conv1D(384, 128, 1×1, 1, 0, 1)
	rd_layer10	rd_layer7, rd_layer9, ad_layer7	$384 \times 64 \times 64$	concatenate(rd_layer7, rd_layer9, ad_layer7)
	rd_layer11	rd_layer10	$128 \times 128 \times 64$	ConvTranspose2D(384, 128, 2×1, (2, 1), 1, 1)
	rd_layer12	rd_layer11	$128 \times 128 \times 64$	Conv2D(128, 128, 3×3, 1, 1, 1) + BN2D(128) + LeakyReLU(0.01) Conv2D(128, 128, 3×3, 1, 1, 1) + BN2D(128) + LeakyReLU(0.01)
	rd_layer13	rd_layer12	$128 \times 256 \times 64$	ConvTranspose2D(128, 128, 2×1, (2, 1), 1, 1)
	rd_layer14	rd_layer13	$128 \times 256 \times 64$	Conv2D(128, 128, 3×3, 1, 1, 1) + BN2D(128) + LeakyReLU(0.01) Conv2D(128, 128, 3×3, 1, 1, 1) + BN2D(128) + LeakyReLU(0.01)
	rd_layer15	rd_layer14	$K \times 256 \times 64$	Conv1D(128, K, 1×1, 1, 0, 1)
RA Decoder	ra_layer9	layer8	$128 \times 64 \times 64$	Conv1D(384, 128, 1×1, 1, 0, 1)
	ra_layer10	ra_layer7, ra_layer9, ad_layer7	$384 \times 64 \times 64$	concatenate(ra_layer7, ra_layer9, ad_layer7)
	ra_layer11	ra_layer10	$384 \times 128 \times 128$	ConvTranspose2D(128, 128, 2×2, 2, 1, 1)
	ra_layer12	ra_layer11	$128 \times 128 \times 128$	Conv2D(128, 128, 3×3, 1, 1, 1) + BN2D(128) + LeakyReLU(0.01) Conv2D(128, 128, 3×3, 1, 1, 1) + BN2D(128) + LeakyReLU(0.01)
	ra_layer13	ra_layer12	$128 \times 256 \times 256$	ConvTranspose2D(128, 128, 2×2, 2, 1, 1)
	ra_layer14	ra_layer13	$128 \times 256 \times 256$	Conv2D(128, 128, 3×3, 1, 1, 1) + BN2D(128) + LeakyReLU(0.01) Conv2D(128, 128, 3×3, 1, 1, 1) + BN2D(128) + LeakyReLU(0.01)
	ra_layer15	ra_layer14	$K \times 256 \times 256$	Conv1D(128, K, 1×1, 1, 0, 1)

Table B.2: **Temporal multi-view network with ASPP modules (TMVA-Net) architecture**. Details about the parameters of each operation are provided in Sec. B.1.2. Let K be the number of classes. The number of input channels in the first layer is fixed to 1 (frames are considered as a sequence), here $q = 4$, thus the number of channels is 5.

View	Method	Param. #	q	Batch size	LR	LR step	Epoch #
RD	FCN-8s [Long *et al.* 2015]	134.3	0	20	10^{-4}	10	100
	U-Net [Ronneberger *et al.* 2015]	17.3	3	6	10^{-4}	20	150
	DeepLabv3+ [Chen *et al.* 2018b]	59.3	3	20	10^{-4}	20	150
	RSS-Net	10.1	3	6	10^{-3}	10	100
	RAMP-CNN	106.4	9	2	10^{-5}	20	150
	MV-Net (ours-baseline)	2.4*	3	13	10^{-4}	20	300
	TMVA-Net (ours)	5.6*	5	6	10^{-4}	20	300
RA	FCN-8s [Long *et al.* 2015]	134.3	0	10	10^{-4}	10	100
	U-Net [Ronneberger *et al.* 2015]	17.3	3	6	10^{-4}	20	150
	DeepLabv3+ [Chen *et al.* 2018b]	59.3	3	20	10^{-4}	20	150
	RSS-Net	10.1	3	6	10^{-4}	10	100
	RAMP-CNN	106.4	9	2	10^{-5}	20	150
	MV-Net (ours-baseline)	2.4*	3	13	10^{-4}	20	300
	TMVA-Net (ours)	5.6*	5	6	10^{-4}	20	300

Table B.3: **Hyper-parameters used for training**. The number of trainable parameters (in millions) for each method corresponds to a single view-segmentation model; Two such models, one for each view, are required for all methods but ours. In contrast, the number of parameters reported for our methods ('*') corresponds to a single model that segments both RD and RA views. RSS-Net and RAMP-CNN have been modified to be trained on both tasks (see Section 5.1.3.3 of the main article). The input of a model consists in $q+1$ successive RAD frames, where q is the number of considered past frames, if any. The learning rate ('LR') step is in epochs.

starts from an already explored weight initialisation to take into account the stochasticity of the optimization. The variability is quantified by considering the average and standard deviation of the performances over the CARRADA-Test dataset regarding the four models for each method and each RADAR view. The variability of the quantitative results by method is illustrated in Figures B.3 and B.4 for the mDice and mIoU evaluation metrics respectively. The TMVA-Net architecture trained with the proposed combination of losses (wCE+SDice+CoL) reached the best average performances with the lower standard deviation, and thus the most stable results, for both RD and RA views, and both mDice and mIoU metrics.

B.1.6 Variability of the loss ablation study

This section analyses the variability of the ablation study comparing the combination of loss functions to train the proposed multi-view architectures. The TMVA-Net architecture has been optimised considering either the CE or the wCE loss function, combined with the SDice or CoL, or both. Details on the loss functions are provided in Section 5.1.2.4. Each model has been trained four times using the CARRADA-Train and CARRADA-Val datasets following the procedures explained in Section 5.1.3. Three of the four models have been trained with different weight initialisation. The training process of the fourth starts from an already explored weight initialisation to take into account the stochasticity of the optimization. The variability is quantified by considering the average and standard deviation of the performances over the CARRADA-Test dataset regarding the four models

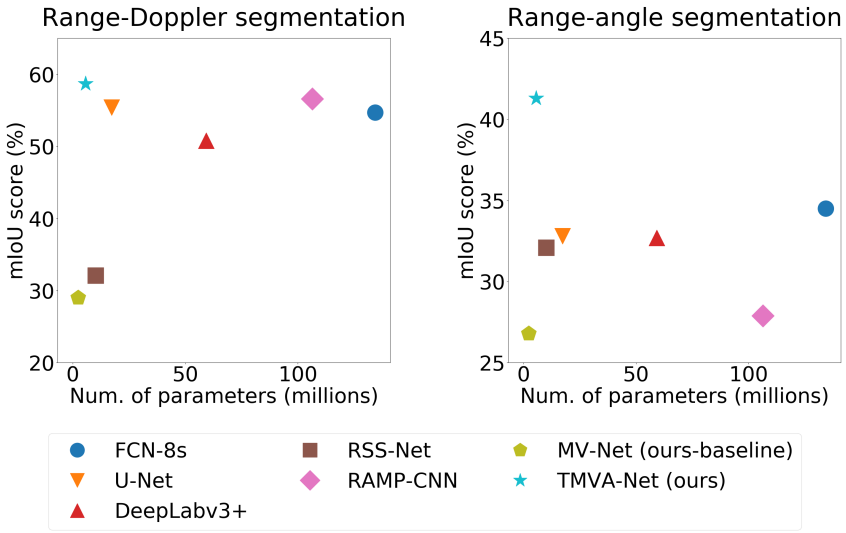


Figure B.2: **Performance-*vs*.-complexity plots for all methods in Range-Doppler (RD) and Range-Angle (RA) tasks**. Performance is assessed by mean Intersection over Union (mIoU) (%) and complexity by the number of parameters (in millions) *for a single task*. Top-left models correspond to the best performing and the lightest. Only our models, MV-Net and TMVA-Net, are able to segment both views simultaneously. For all the other methods, two distinct models must be trained to address both tasks, which doubles the number of actual parameters.

for each method and each RADAR view. The variability of the loss ablation study usng the TMVA-Net architecture is illustrated in Figure B.5 for the mDice evaluation metric. The average performances increased for both RD and RA segmentation by integrating the CoL term to each individual loss term of CE, wCE and SDice[1]. It indicates that the CoL term has a significant impact on the training process. The combination wCE+SDice+CoL reached the best trade-off between performances and stability of the results for both RD and RA.

B.1.7 Qualitative results on CARRADA

Additional qualitative results are shown in Figure B.6 for each method on scenes (1-2) from the CARRADA-Test. For the scene (1), RAMP-CNN (g) and TMVA-Net (i-j) display well segmented RD views. However, only TMVA-Net with CoL (j) is able to localise and classify both objects in the RD and RA views of the first example. In scene (2), four methods (d-e-i-j) are able to well localise objects in the RD view. Once again, only TVMA-Net with CoL (j) is able to well segment objects in both RD and RA views while our method without CoL (i) predicts pedestrian and cyclist categories for pixels of the same object.

[1]Except for the combination SDice+CoL for the RD view.

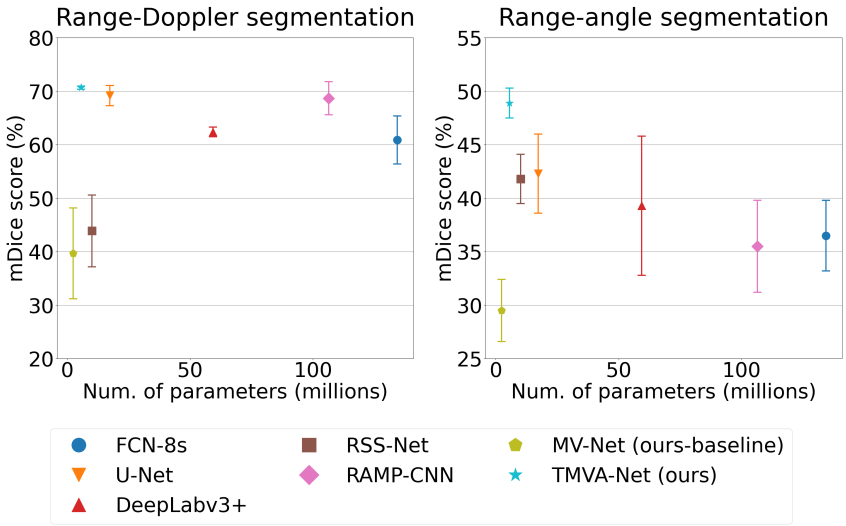


Figure B.3: **Variability of the performances-*vs*.-complexity plots for all methods in Range-Doppler (RD) and Range-Angle (RA) tasks regarding the mDice metric.**. Performance is assessed by mDice (%) and complexity by the number of parameters (in millions) *for a single task*. Top-left models correspond to the best performing and the lightest. The variability of the performances consists in evaluating the average (point) and standard deviation (vertical line) of four trained models. Only the proposed models, MV-Net and TMVA-Net, are able to segment both views simultaneously. For all the other methods, two distinct models must be trained to address both tasks, which doubles the number of actual parameters.

B.1.8 Qualitative results on RADDet

Figure B.7 shows qualitative results of two complex urban scenes (1-2) for each method trained on RADDet-Train and RADDet-Val, and tested on RADDet-Test. For the scene (1), DeepLabv3+ (e), RSS-Net (f) and RMVA-Net (i) display well segmented RD views. However, only the TMVA-Net trained with the proposed loss combination (wCE+SDice+CoL) is able to localise and classify two cars and a pedestrian in the RD and RA views of the first example. In scene (2), most of the methods successfully segment four cars in both RD and RA views. However, only the TMVA-Net is able to localise and classify the pedestrian in the RD view. It is well localised on the RA but misclassified as a car instead of pedestrian. The annotated shape of the pedestrian is small and thus difficult to recognize. A reformulation of CoL could be explored to better enforce the spatial coherence.

B.1.9 Qualitative results on in-house dataset

Figure B.8 shows qualitative results for each method trained on CARRADA-Train and CARRADA-Val, and tested on in-house sequences of complex urban scenes (1-2) with a different range resolution. The qualitative examples and results have been cropped with respect to the minimum and maximum range of the dataset used for training. The ground-

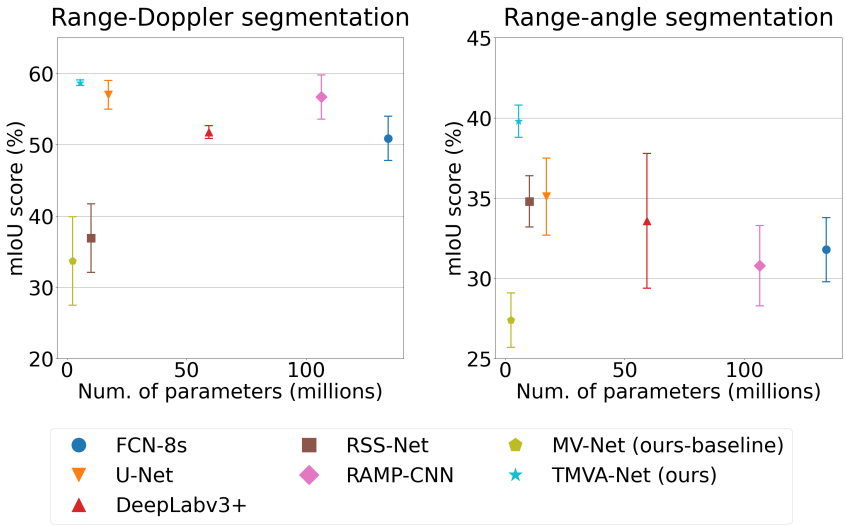


Figure B.4: **Variability of the performances-*vs*.-complexity plots for all methods in Range-Doppler (RD) and Range-Angle (RA) tasks regarding the mIoU metric.**. Performance is assessed by mean Intersection over Union (%) and complexity by the number of parameters (in millions) *for a single task*. Top-left models correspond to the best performing and the lightest. he variability of the performances consists in evaluating the average (point) and standard deviation (vertical line) of four trained models. Only our models, MV-Net and TMVA-Net, are able to segment both views simultaneously. For all the other methods, two distinct models must be trained to address both tasks, which doubles the number of actual parameters.

truth masks in columns (1-b) and (2-b) are empty because the RADAR views are not annotated. In scene (1), only TMVA-Net models (i-j) are able to localise and classify the signals related to the pedestrians and cars in both the RD and the RA views. In scene (2), only TMVA-Net (i-j) methods succeed in localising and classifying cars and pedestrians in the RA view. We note that TMVA-Net without CoL (i) detects more car signals while TMVA-Net with CoL (j) is the only method capable of distinguishing pedestrian signatures on both RD and RA views.

These two examples in complex urban scenes suggest that our method has successfully learnt object signatures in the CARRADA dataset and is able to generalise well.

B.2 Sensor fusion

B.2.1 Sensor settings of the nuScenes dataset

Figure B.9 illustrates the sensor setup used to record the nuScenes dataset [Caesar *et al.* 2020]. The orientation axis is specific to each sensor requiring transformations to manipulate both RADAR and LiDAR point clouds.

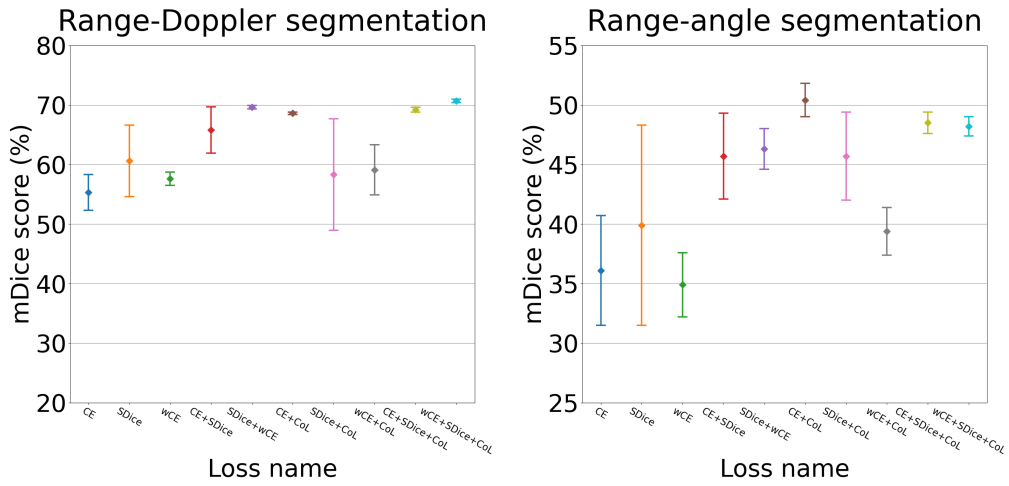


Figure B.5: **Variability of the performances plots for all combination of losses in Range-Doppler (RD) and Range-Angle (RA) tasks regarding the mDice metric.**. Performance is assessed by mDice (%). All the possible combination of losses are considered, either with the Cross-Entropy (CE) or weighted Cross-Entropy (wCE) loss, mixed with or without the Soft Dice (SDice) and Coherence Loss (CoL) losses. The performance variability consists in evaluating the mean (point) and standard deviation (vertical line) of four trained models. The proposed combination (wCE+SDice+CoL) is the best trade-off between performance and stability on both RD and RA.

B.2.2 Qualitative results of the propagation and fusion module

Figure B.10 presents additional qualitative results of our proposed propagation and fusion module on a scene of the nuScenes dataset. Our proposed method detailed in Section 5.2.2 succeed to obtain an enriched point cloud with a denser Doppler information.

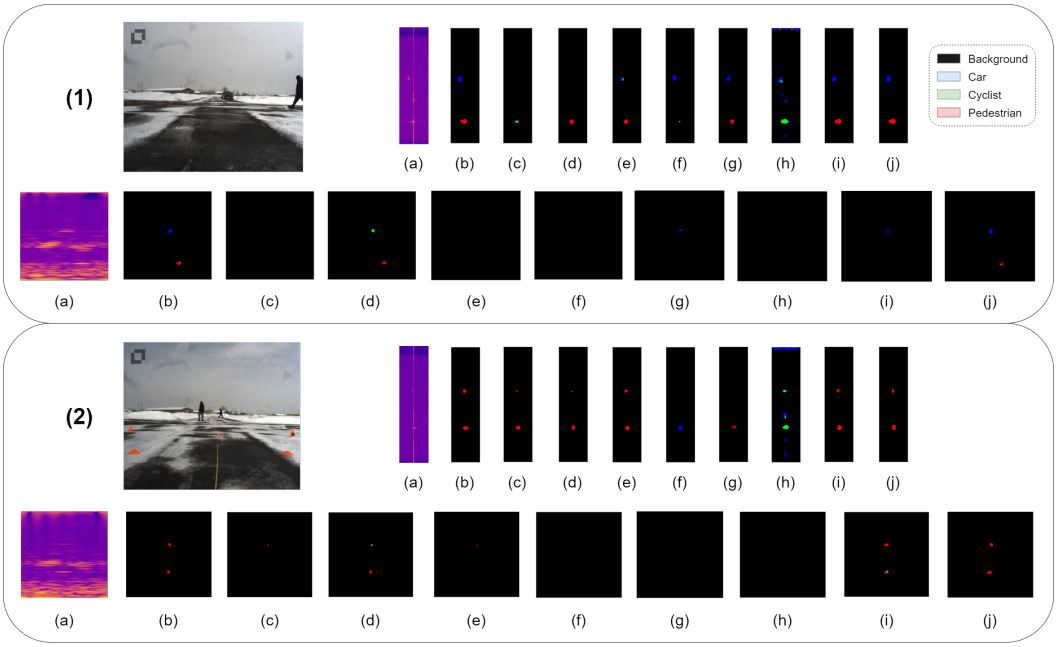


Figure B.6: **Qualitative results on two test scenes of CARRADA-Test**. (1) and (2) are two independent examples. (*Top*) camera image of the scene and results of the Range-Doppler segmentation; (*Bottom*) Results of the Range-Angle Segmentation. (a) RADAR view signal, (b) ground-truth mask, (c) FCN8s, (d) U-Net, (e) DeepLabv3+, (f) RSS-Net, (g) RAMP-CNN, (h) MV-Net (our baseline w/ wCE+SDice loss), (i) TMVA-Net (ours, w/ wCE+SDice loss), (j) TMVA-Net (ours, w/ wCE+SDice+CoL loss).

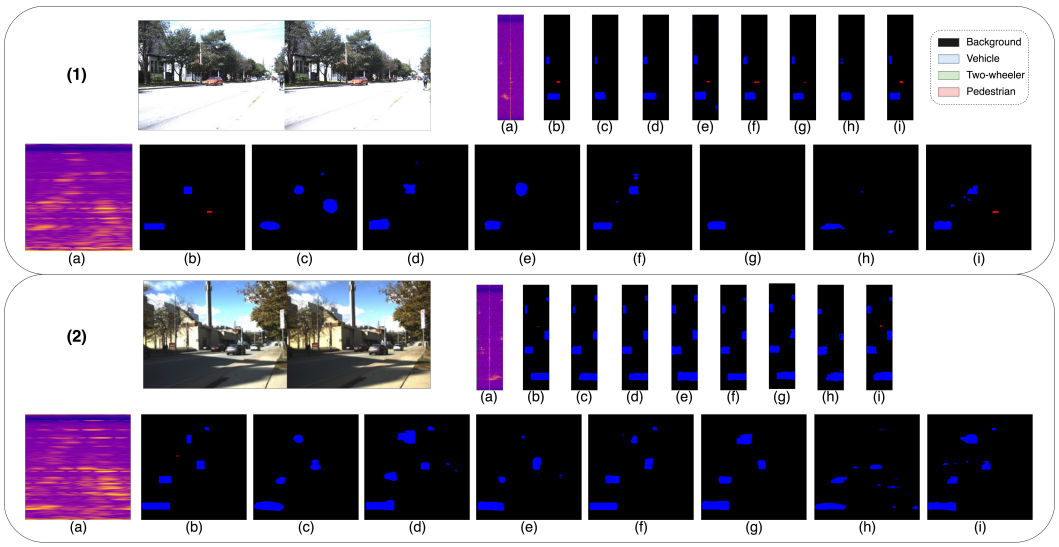


Figure B.7: **Qualitative results on a test scene of RADDet**. (*Top*) camera image of the scene and results of the Range-Doppler segmentation; (*Bottom*) Results of the Range-Angle Segmentation. (a) Radar view signal, (b) ground-truth mask, (c) FCN8s, (d) U-Net, (e) DeepLabv3+, (f) RSS-Net, (g) RAMP-CNN, (h) MV-Net (our baseline w/ wCE+SDice+CoL loss), (i) TMVA-Net (ours, w/ wCE+SDice+CoL loss).

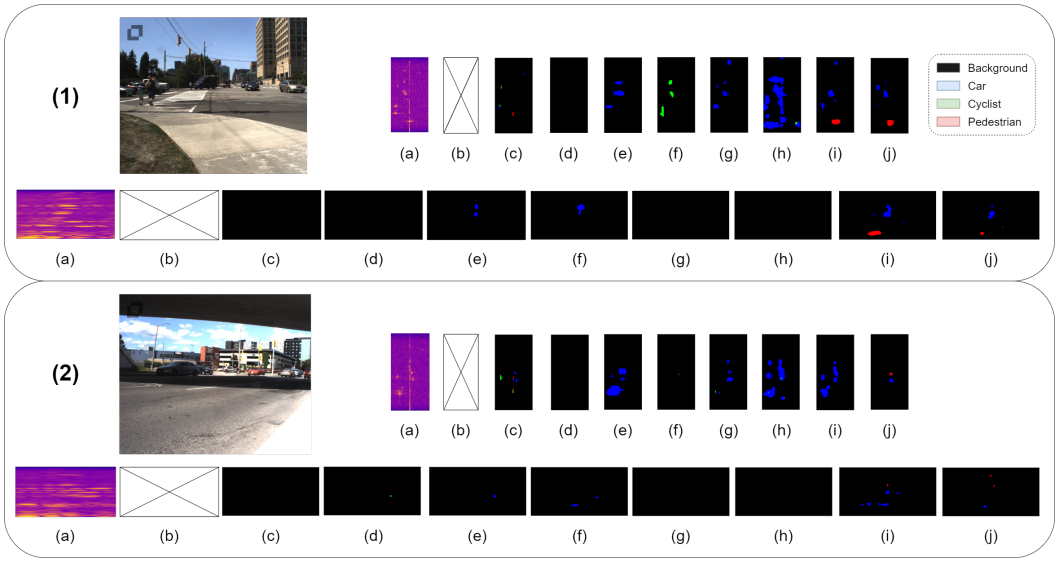


Figure B.8: **Qualitative results on two test scenes of in-house sequences**. (1) and (2) are two independent examples. (*Top*) camera image of the scene and results of the Range-Doppler segmentation; (*Bottom*) Results of the Range-Angle Segmentation. (a) RADAR view signal, (b) ground-truth mask, (c) FCN8s, (d) U-Net, (e) DeepLabv3+, (f) RSS-Net, (g) RAMP-CNN, (h) MV-Net (our baseline w/ wCE+SDice loss), (i) TMVA-Net (ours, w/ wCE+SDice loss), (j) TMVA-Net (ours, w/ wCE+SDice+CoL loss).

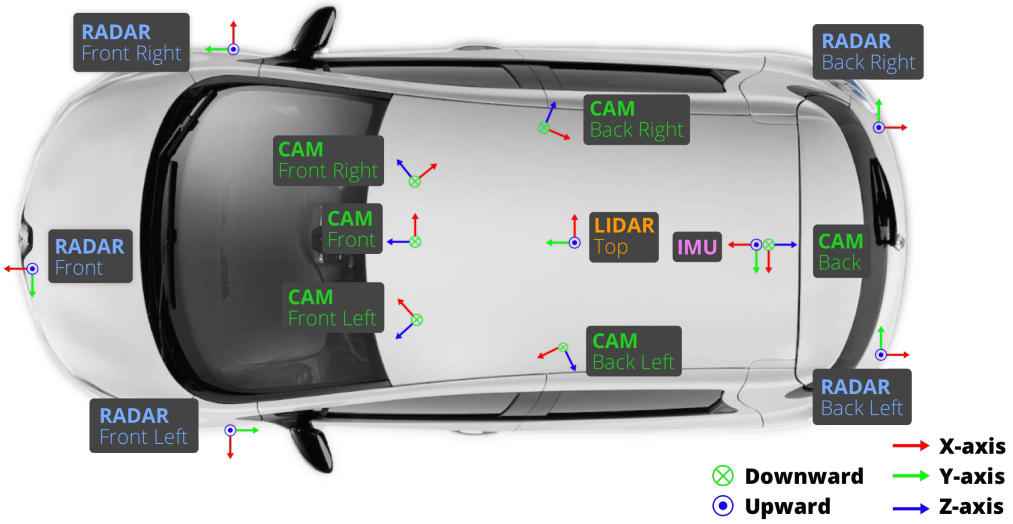


Figure B.9: **Qualitative results on two test scenes of in-house sequences**. Source: [Caesar *et al.* 2020].

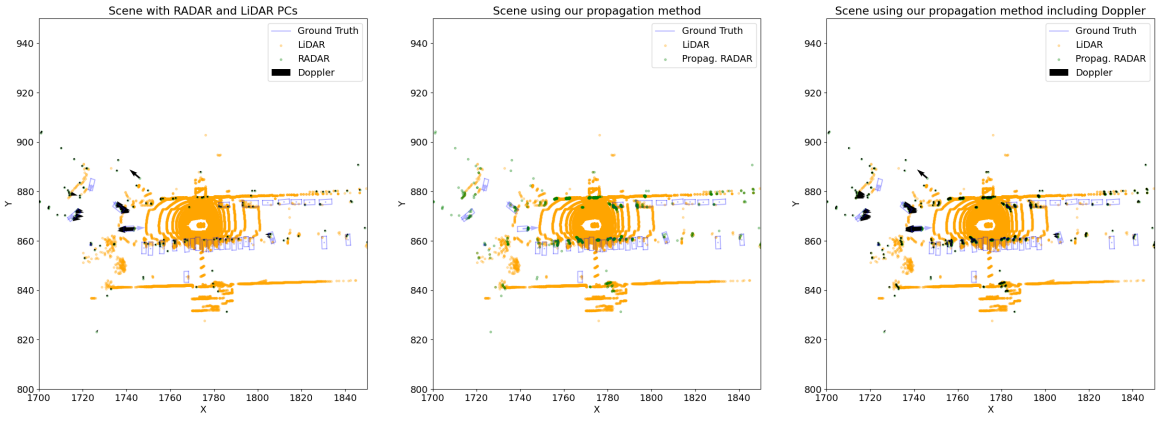


Figure B.10: **Qualitative results on the nuScenes dataset of our propose propagation and fusion module**. (Left) Scene in Bird's Eye View representation with LiDAR and RADAR point clouds. (Middle) The point cloud illustrated in green groups the RADAR and propagated RADAR points with Doppler and reflectivities. (Right) The propagated Doppler information is illustrated with black arrows to distinguish moving objects.

APPENDIX C

High-definition RADAR

C.1 Ablation study of the MIMO pre-encoder

The role of the MIMO pre-encoder is to de-interleave the Range-Doppler (RD) and to transform them into a representation that is compact and still allows, through learning, the prediction of azimuth angles along with other information on reflectors. The input of the MIMO pre-encoder is composed of the $N_{\text{Rx}} = 16$ RD in complex numbers, one for each Rx. The real and imaginary parts are stacked, yielding an input tensor of total size $B_{\text{R}} \times B_{\text{D}} \times 2N_{\text{Rx}}$, *i.e.*, $512 \times 256 \times 32$. The ablation study consists in evaluating the performance of FFT-RadNet's detection head while reducing the number of feature channels that the MIMO pre-encoder outputs. The maximum number of output channels is the number of virtual antennas with a complex signal (real and imaginary parts), *i.e.*, $N_{\text{Tx}} \cdot 2 \cdot N_{\text{Rx}} = 384$. We vary the number of output channels from a minimum of 24 to this maximum value and compute the detection performance on the validation set. The results of this ablation study are reported in Figure C.1. We measure the detection performance with the F1-score, classically defined as F1-score $= \frac{\text{AP} \cdot \text{AR}}{\text{AP} + \text{AR}}$, which aggregates in a single metric both the Average Precision (AP) and the Average Recall (AR). We observe that the best performance is reached with 192 output channels, hence half of the maximum output size. This compressed output is the one that captures at best the range and azimuth information from the RD inputs toward the detection and segmentation tasks.

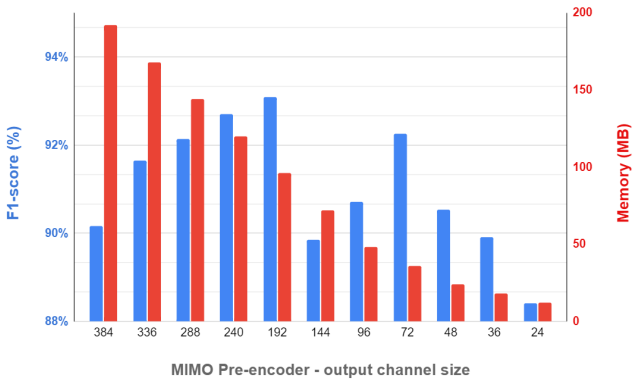


Figure C.1: **MIMO pre-encoder ablation study**. Influence of the number of output channels of the pre-encoder on the memory footprint and the performance of the detection head.

ECOLE DOCTORALE

Titre : Apprentissage profond pour l'exploitation de données radar dans la conduite autonome

Mots clés : Apprentissage profond, données RADAR, traitement du signal, vision par ordinateur

Résumé : La conduite autonome exige une compréhension détaillée de scènes de conduite complexes. La redondance et la complémentarité des capteurs du véhicule permettent une compréhension précise et robuste de l'environnement, augmentant ainsi le niveau de performance et de sécurité. Cette thèse se concentre sur le RADAR automobile, qui est un capteur actif à faible coût mesurant les propriétés des objets environnants, y compris leur vitesse relative, et qui a l'avantage de ne pas être affecté par des conditions météorologiques défavorables. Avec les progrès rapides de l'apprentissage profond et la disponibilité d'ensembles de données publiques sur la conduite, la capacité de perception des systèmes de conduite basés sur la vision (par exemple, la détection d'objets ou la prédiction de trajectoire) s'est considérablement améliorée. Le capteur RADAR est rarement utilisé pour la compréhension de scène en raison de sa faible résolution angulaire, de la taille, du bruit et de la complexité des données brutes RADAR ainsi que du manque d'ensembles de données disponibles. Cette thèse propose une étude approfondie de la compréhension de scènes RADAR, de la construction d'un jeu de données annotées à la conception d'architectures d'apprentissage profond adaptées. Tout d'abord, cette thèse détaille des approches permettant de remédier au manque de données. Une simulation simple ainsi que des méthodes génératives pour créer des données annotées seront présentées. Elle décrit également le jeu de données CARRADA, composé de données synchronisées de caméra et de RADAR avec une méthode semi-automatique générant des annotations sur les représentations RADAR. Cette thèse présente ensuite un ensemble d'architectures d'apprentissage profond avec leurs fonctions de perte associées pour la segmentation sémantique RADAR. Elle décrit également une méthode permettant d'ouvrir la recherche sur la fusion des capteurs LiDAR et RADAR pour la compréhension de scènes. Enfin, cette thèse expose une contribution collaborative, le jeu de données RADIal avec RADAR haute définition (HD), LiDAR et caméra synchronisés. Une architecture d'apprentissage profond est également proposée pour estimer le pipeline de traitement du signal RADAR tout en effectuant simultanément un apprentissage multitâche pour la détection d'objets et la segmentation de l'espace libre de conduite.

Title : Deep learning for radar data exploitation of autonomous vehicle

Keywords : Deep learning, RADAR data, signal processing, computer vision

Abstract : Autonomous driving requires a detailed understanding of complex driving scenes. The redundancy and complementarity of the vehicle's sensors provide an accurate and robust comprehension of the environment, thereby increasing the level of performance and safety. This thesis focuses the on automotive RADAR, which is a low-cost active sensor measuring properties of surrounding objects, including their relative speed, and has the key advantage of not being impacted by adverse weather conditions. With the rapid progress of deep learning and the availability of public driving datasets, the perception ability of vision-based driving systems (e.g., detection of objects or trajectory prediction) has considerably improved. The RADAR sensor is seldom used for scene understanding due to its poor angular resolution, the size, noise, and complexity of RADAR raw data as well as the lack of available datasets. This thesis proposes an extensive study of RADAR scene understanding, from the construction of an annotated dataset to the conception of adapted deep learning architectures. First, this thesis details approaches to tackle the current lack of data. A simple simulation as well as generative methods for creating annotated data are presented. It also describes the CARRADA dataset, composed of synchronised camera and RADAR data with a semi-automatic method generating annotations on the RADAR representations. This thesis then presents a proposed set of deep learning architectures with their associated loss functions for RADAR semantic segmentation. The proposed architecture with the best results outperforms alternative models, derived either from the semantic segmentation of natural images or from RADAR scene understanding, while requiring significantly fewer parameters. It also introduces a method to open up research into the fusion of LiDAR and RADAR sensors for scene understanding. Finally, this thesis exposes a collaborative contribution, the RADIal dataset with synchronised High-Definition (HD) RADAR, LiDAR and camera. A deep learning architecture is also proposed to estimate the RADAR signal processing pipeline while performing multitask learning for object detection and free driving space segmentation simultaneously.

Institut Polytechnique de Paris
91120 Palaiseau, France

ECOLE DOCTORALE