UNIVERSITY OF OTTAWA

MASTER'S THESIS

Object Detection from FMCW Radar Using Deep Learning

Author:
Ao Zhang

Supervisor:
Prof. Robert LAGANIERE

uOttawa

A thesis submitted in partial fulfillment of the requirements for the degree of Master of Applied Science

in the

School of Electrical Engineering and Computer Science
Faculty of Engineering, University of Ottawa

Declaration of Authorship

I, Ao Zhang, declare that this thesis titled, "Object Detection from FMCW Radar Using Deep Learning" and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Date: 27^{th} April, 2021

UNIVERSITY OF OTTAWA

Abstract

Faculty of Engineering
School of Electrical Engineering and Computer Science

Master of Applied Science

Object Detection from FMCW Radar Using Deep Learning

by Ao Zhang

Sensors, as a crucial part of autonomous driving, are primarily used for perceiving the environment. The recent deep learning development of different sensors has demonstrated the ability of machines recognizing and understanding their surroundings.

Automotive radar, as a primary sensor for self-driving vehicles, is well-known for its robustness against variable lighting and weather conditions. Compared with camera-based deep learning development, Object detection using automotive radars has not been explored to its full extent. This can be attributed to the lack of public radar datasets.

In this thesis, we collect a novel radar dataset that contains radar data in the form of Range-Azimuth-Doppler tensors along with the bounding boxes on the tensor for dynamic road users, category labels, and 2D bounding boxes on the Cartesian Bird-Eye-View range map. To build the dataset, we propose an instance-wise auto-annotation algorithm. Furthermore, a novel Range-Azimuth-Doppler based multi-class object detection deep learning model is proposed. The algorithm is a one-stage anchor-based detector that generates both 3D bounding boxes and 2D bounding boxes on Range-Azimuth-Doppler and Cartesian domains, respectively. Our proposed algorithm achieves 56.3% AP with IOU of 0.3 on 3D bounding box predictions, and 51.6% with IOU of 0.5 on 2D bounding box predictions. Our dataset and the code can be found at `https://github.com/ZhangAoCanada/RADDet.git`.

Keywords: Autonomous Driving, Self-driving, Automotive Radar, Range, Azimuth, Doppler, Cartesian, Auto-annotation, Radar Dataset, Deep Learning, Object Detection

Acknowledgements

I would like to thank my supervisor, Prof. Robert Laganiere, for giving me this wonderful opportunity to perform this research. His constant support and valuable feedback always helped me overcome the difficulties whenever the problems stopped this research being continued. Without his guidance, this thesis would have never been completed.

My deep appreciation also goes to Erlik Nowruzi, Fahed Hassanat, Prince Kapoor, Dhanvin Kolhatkar, Pascal Blais and all those who are working in Sensor Cortek. Their high-quality data capture tools and professional experience ensure this thesis being conducted expectedly.

This thesis would have been much harder without the support of my friends in the VIVA lab. I would like to thank Wassim Ahmar, Hamed Aghdam, Rytis Verbickas, Md Atiqur Rahman, Wen Wen.

And finally, I express my deepest thanks to my parents who support me financially and mentally. Being studying in another country, I would not have finished this program if without their encouragement and care.

List of Publications

RADDet: Range-Azimuth-Doppler based Radar Object Detection for Dynamic Road Users, *Ao Zhang, Farzan Erlik Nowruzi, Robert Laganiere, 18th Conference on Robots and Vision, CRV 2021 - Burnaby, British Columbia*

Contents

List of Figures

List of Tables

List of Abbreviations

ADC	**A**nalog to **D**igital **C**onverter
API	**A**pplication **P**rogramming **I**nterface
BP	**B**ack **P**ropagation
CFAR	**C**onstant **F**alse **A**larm **R**ate
CNN	**C**onvolutional of **N**eural **N**etworks
CUT	**C**ell **U**nder **T**est
CV	**C**omputer **V**ision
DBSCAN	**D**ensity-**B**ased **S**patial **C**lustering of **A**pplications with **N**oise
DL	**D**eep **L**earning
DOA	**D**irection **o**f **A**rrival
DoF	**D**egree **o**f **F**reedom
DSP	**D**igital **S**ignal **P**rocessing
FFT	**F**ast **F**ourier **T**ransform
FMCW	**F**requency **M**odulated **C**ontinuous **W**ave
FOV	**F**ield **O**f **V**iew
FPS	**F**rames **P**er **S**econd
GPS	**G**lobal **P**ositioning **S**ystem
GPU	**G**raphic of **P**rocessing **U**nit
IF	**I**ntermediate **F**requency
IMU	**I**nertial **M**easurement **U**nit
IoU	**I**ntersection **o**f **U**nion
LiDAR	**Li**ght **D**etection **A**nd **R**anging
mAP	**m**ean **A**verage **P**recision
MIMO	**M**ultiple **I**nput **M**ultiple **O**utput
ML	**M**achine **L**earning
MLP	**M**ulti-**L**ayer **P**erceptron
NAS	**N**eural **A**rchitecture **S**earch
NLP	**N**atural **L**anguage **P**rocessing
RA	**R**ange **A**zimuth
RAD	**R**ange **A**zimuth **D**oppler
RCS	**R**adar **C**ross **S**ection
RD	**R**ange **D**oppler
ReLU	**R**ectified of **L**inear **U**nit
RL	**R**einforcement **L**earning
ROS	**R**obotics **O**perating **S**ystem
SGBM	**S**emi-**G**lobal **B**lock **M**atching
SGD	**S**tochastic of **G**radient **D**escent
ToF	**T**ime **o**f **F**light

Chapter 1

Introduction

Autonomous driving, as a very popular research topic, has been widely studied in recent years. Major vehicle manufacturers such as Tesla [1] and Mercedes-Benz [2] have invested millions of dollars into this field for the development of self-driving cars. With the help of recent up-rising technologies such as Machine Learning (ML), self-driving vehicles have become more and more stable and reliable. Modern autonomous driving was first introduced by the ALV project [31] at Carnegie Mellon University, which was funded by the Defense Advanced Research Projects Agency (DARPA), in 1984 [82]. Since then, the increasing popularity of autonomous vehicles has drawn public attention, and rapid improvements in this field have been witnessed over the last decade. As a result, the impressive achievement obtained has stimulated more studies in this field, and thus more targeted improvements have been introduced. Today, with the development of commercial self-driving vehicles such as the Tesla Model Series, autonomous driving is becoming a part of our lives.

One of the biggest concerns in the deployment of autonomous vehicles is safety. Many related accidents have been reported by the major media recently. Driver fatality cases such as [85], [77] and [78], and pedestrian fatality cases such as [42] were first brought into public in 2018. After that, the increasing public concerns have posed many questions about self-driving cars being deployed into our lives. Apart from the rising urgency of the legislation for autonomous driving, researchers are also seeking a solution to improve the safety of autopilot vehicles. The objective of the safety improvement is to make the autonomous agent travel reliably and safely through its environment by properly sensing, understanding, and judging its surroundings [82]. Therefore, the core technologies, sensors, have become the focus of recent studies in autonomous driving.

Sensors, known as the eyes of autopilot vehicles, are primarily used for perceiving the environment. Different sensors such as LiDAR, Radar, Camera, Sonar, GPS and IMU, have been employed in a wide variety of researches and applications. Based on their different properties, sensors are generally split into two categories, 3D sensors and 2D sensors. 3D sensors such as Lidar and stereo cameras, can provide 3D information

[1] https://www.tesla.com/
[2] https://www.mercedes-benz.com/

of the surroundings in a format known as point-cloud. Their accurate sensing has contributed to the improvement of many key components in autonomous driving, such as navigation systems and 3D reconstruction. Despite that, the applications of these 3D sensors are still very limited due to their high prices and excessive computational consumption. 2D sensors, on the other hand, are normally cheaper than 3D sensors and thus widely used in various applications. Camera, as a typical 2D sensor, provides perspective views of the surroundings and presents them into images. Since this type of interpretation is more intuitive to human beings, a great number of researches have been conducted with cameras, especially in the field of machine learning. The recent machine learning development in autonomous driving has shown the capability of the computer to accurately understand and localize its surroundings through images. Other types of 2D sensors, including radar and sonar, estimates the positioning information of the surroundings by using Time of Flight (ToF) technology with specially designed signals. In this research, we focus on machine learning development for a specific type of 2D sensor, Automotive radar.

1.1 Overview of Automotive Radar

Automotive radar, also known as Frequency Modulated Continuous Wave (FMCW) radar, is one of the primary sensors used in autonomous vehicles. Its robustness against various lighting and weather conditions is considered as a necessary feature for modern self-driving cars. The low cost of automotive radars also makes them available for many other civilian applications such as traffic surveillance.

The first practical application of FMCW radar started in 1928, when J.O. Bentlei filed the American patent on an "airplane altitude indicating system" [33]. Since then, FMCW radars had been widely used for military purposes. The idea of automotive radars being used in civilian applications began in the early 1940s, when the ultrahigh-frequency band was exploited. Most of the theoretical works on FMCW radar were published during a period from the late 1940s to the early 1960s [33]. Few years after that, the applications of this type of technology started to be seen in various civilian applications such as navigational radar and vehicle collision warning systems.

The basic function of automotive radar is accomplished by ToF technology. The designed signals with modulated frequencies are transmitted from the radar and reflected by the objects before being received by the radar again. The distances between the radar and the objects are calculated based on the time duration between the transmitted signals and received signals. With the help of Doppler Effect, the velocities of those objects can also be computed with the frequency differences between the signals. Modern FMCW radars are usually designed with Multiple Inputs and Multiple Outputs (MIMO), shown in Figure 1.1. The differences between the signals observed from different onboard receivers also enable the computation of the angles of received signals. Therefore, FMCW radar is capable of detecting both positions and speeds of its

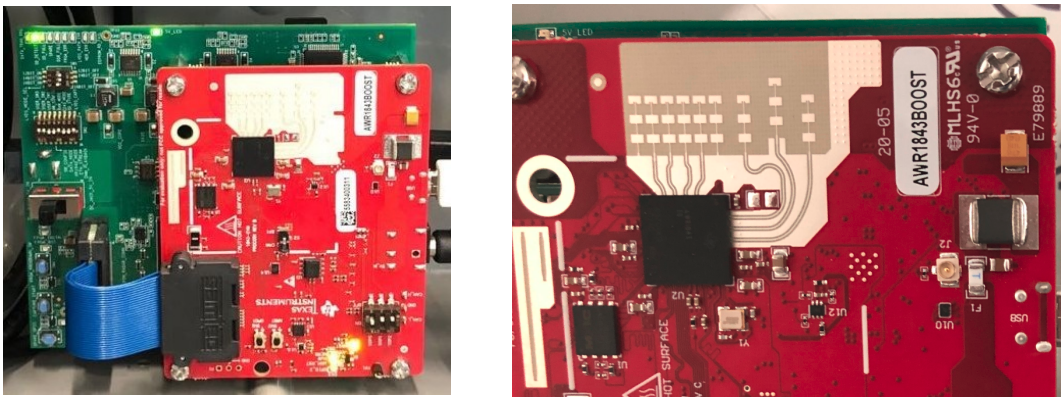

FIGURE 1.1: Left: AWR1843Boost FMCW mmWave Radar with DCA1000EVM Data Capture module of this project. Right: AWR1843Boost antennas arrangement.

surroundings. On the other hand, since the transmitted signals are normally designed with ultra-high frequencies, they are able to penetrate through certain objects such as water and glass. As a result, different weather conditions such as fog and rain, can hardly affect the function of FMCW radars. The advantages of automotive radars over other sensors can be listed as follow:

- Able to detect the positions and the velocities of different targets simultaneously.
- Robust against variable lighting and weather conditions such as fog, rain and snow.
- Highly configurable with various available range/doppler profiles.
- Light-weight, low-price and small energy consumption.

Although FMCW radar has many advantages, there are still a number of challenges that need to be addressed. The main issue with radar is the high level of noise found in its signals. The variability of the reflectivities of different object surfaces also represents a challenge for detections. Various algorithms such as Constant False Alarm Rate (CFAR), have been developed over many years to overcome the problem. Recent technologies such as machine learning are also studied for improving the detection quality of FMCW radars.

1.2 Overview of Object Detection

Deep Learning (DL), as a sub-field of machine learning, has been one of the most intensively researched topics in recent years. Its idea is to teach the machine to perform a specific task through the data. In Computer Vision (CV), tasks such as object detection, instance segmentation and super-resolution, have been widely studied and applied to our daily lives. Among them, object detection is considered as one of the most important branches of autonomous driving due to its high precision and fast speed.

Modern object detection development started in the early 2010s, with the introduction of a special type of deep networks, Convolutional Neural Networks (CNNs). The proposal of simultaneously recognizing and localizing the objects in images rapidly raised the interests of researchers in this field. With the development of object detection algorithms such as R-CNN [20], YOLO [54] and SSD [41], applications started to be brought into various areas such as autonomous driving, machine inspection and face detection. As deep learning algorithms keep evolving, more advanced object detection algorithms with higher performance and faster speeds have been developed during the most recent years.

The objective of object detection is to use deep learning models to detect not only the positions but also the categories of the targeted objects in the images. As there is normally more than one object appearing in an image, a technique called bounding box is used to specify the locations of different objects. Figure 1.2 shows a sample of results produced by a modern object detection algorithm.

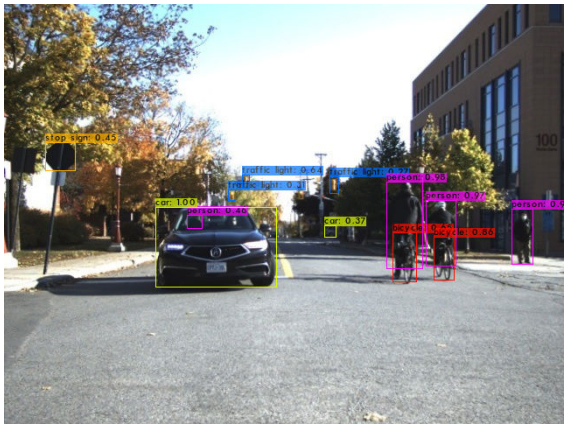

FIGURE 1.2: A sample of image object detection using YOLOv4 [2].

The key contribution to the modern development of image object detection is the introduction of public datasets. These datasets are normally categorized into generic datasets and task-targeted datasets. Generic datasets such as COCO [39] and ImageNet [60] provide thousands of images with millions of object instances and hundreds of categories to all researchers for deep learning development on various tasks, including object detection, semantic segmentation, object tracking and etc. Huge datasets often help to improve the generalization of the corresponding development. Task-targeted datasets, on the other hand, are normally designed for the specific tasks. For example, KITTI [17] dataset, which is developed specifically for autonomous driving, contains the data from various sensors, such as Lidar and Stereo cameras. The object categories in the dataset are also limited to the self-driving related classes such as pedestrians, cars and trucks. This type of dataset can provide more insights into a specific area and help the improvement of task-targeted object detection models.

1.3 Problem Statement

As image object detection has shown its capability of achieving high precisions, researchers started to seek the possibilities of deep learning algorithms applied to the

radar domain. Although different applications of radar-based detections have been explored recently, the total number of related researches is still very limited. One primary reason for this phenomenon is the lack of public radar datasets.

Unlike image-based datasets that provide intuitive data interpretations, the generation of radar-based datasets is relatively difficult due to several reasons. First, the highly configurable property of FMCW radars makes it hard for dataset standardization. Different configurations can result in different data resolutions and range limitations. Second, different radar data formats, such as point-cloud format and signal spectrum format, also make it difficult to standardize radar datasets. Third, the low readability of radar data can lead to problems for manual annotations as human beings can hardly recognize the differences between the objects in radar frames. Fourth, the noisy data also complicates the process of dataset generation since it can dramatically impact the annotation quality.

For radar object detection, although recent researches have proven that image-based models such as FCN [67] can be transferred to the radar-based development, the possibility of more common image object detection models being employed in radar object detection has not been fully explored. Therefore, more model explorations and customizations need to be done for the further improvement of radar object detection.

In this research, we introduce a novel publicly available dataset along with our auto-annotation algorithm, and a radar object detection model for dynamic road users. Our contributions are as follows:

- A novel dataset[3] that contains radar data in the form of Range-Azimuth-Doppler (RAD) representations with the corresponding annotations for various object classes is introduced. The dataset is set publicly available. In addition, the automatic annotation method for generating ground-truth labels on all dimensions of RAD tensors and the corresponding Cartesian representations is also provided.

- A new radar object detection model with a ResNet [25]-like backbone and a dual detection head is proposed. Our new dual detection head, with a 3D detection head on RAD data and a 2D detection head on the data in Cartesian coordinates, is developed based on YOLO [6] detection head. Our proposed model is extensively evaluated against the well-known image-based object detection counterparts. Results show that our proposal achieves state-of-the-art performance in the object detection task from radar data.

1.4 Thesis Outline

The remainder of this thesis is organized as follows.

Chapter 1 "Introduction": This chapter introduces an overview of two primary sensors used in autonomous driving, radar and camera, and briefly discusses the problems

[3] https://www.site.uottawa.ca/research/viva/projects/raddet/index.html

of the recent radar object detection development along with our contributions in this research.

Chapter 2 "Background": This chapter introduces the background knowledge of FMCW Signal Processing, Stereo Depth Estimation and image-based Deep Learning Applications.

Chapter 3 "Related Work": This chapter presents a literature review of the most recent development on radar object detection and image object detection.

Chapter 4 "Dataset Generation": This chapter describes the work for generating our proposed radar dataset, including hardware setups, sensor registration and a novel auto-annotation algorithm. A statistical analysis of the dataset is also presented at the end of the chapter.

Chapter 5 "Range-Azimuth-Doppler based Object Detection": This chapter presents our proposed deep learning model, which consists of a backbone and a dual detection head, for radar object detection.

Chapter 6: "Experiments": This chapter provides the information of our model exploration along with the results and analysis of state-of-the-art image-based models on radar object detection, including several popular backbones and self-attention layers.

Chapter 7: "Conclusion and Future Work": This chapter summarizes the proposed auto-annotation algorithm, the radar dataset and our radar object detection model, and proposes the potential improvements that can be done in future research.

Chapter 2

Background

Raw data acquired from Frequency Modulated Continuous Wave (FMCW) radars are unreadable for human observers. To deal with this problem, traditional Digital Signal Processing (DSP) has to be applied to the received raw Analog to Digital Converter (ADC) signals. The outputs of the traditional DSP are normally formulated into Range-Azimuth-Doppler (RAD) data format, which is used for the development of our auto-annotation algorithm and radar dataset. In this chapter, we first introduce the background information of traditional radar DSP. Then, the fundamental of stereo depth estimation is introduced for the reason that our auto-annotation algorithm is built with a FMCW radar and a pair of stereo cameras. Finally, a brief introduction of modern image-based deep learning algorithms is presented at the end of this chapter to provide related background information.

2.1 FMCW radar

Frequency Modulated Continuous Wave (FMCW) radar is a type of radar where a known modulated frequency continuous wave radio energy is transmitted and then received from any reflecting objects [9]. Different types of modulated frequencies such as sawtooth type, triangle type and sinusoidal type, have been applied to various real-world applications. In our research, we focus on sawtooth type signals from a *Texas Instrument* AWR1843BOOST[1] radar for data collection and annotation. Figure 2.1 shows the basic profile of the signals with sawtooth form of instantaneous frequency. The frequency of this type of transmitted signals changes linearly and periodically with respect to time. For convenience, an entire period is often called one "chirp". The time duration of one chirp is also called Sweep Time (T_c). Combined with the built-in properties such as frequency bandwidth (B) and initial frequency (f_0) of our FMCW radar, the mathematical model of transmitted signal frequency is formed as Equation 2.1

$$f(t) = f_0 + \frac{B}{T_c}t, \quad t \in [0, T_c] \tag{2.1}$$

[1] https://www.ti.com/

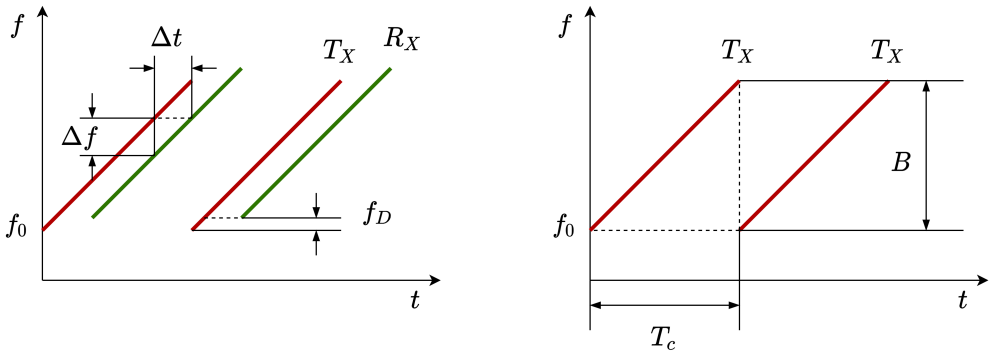

FIGURE 2.1: Left: Sawtooth chirp profile, where the echo delay Δt is transformed to the frequency difference Δf and f_D is the doppler shift. Right: Frequency profile, where T_c is the Sweep Time and B is the frequency bandwidth.

where, t is the time variable. The term $\frac{B}{T_c}$ can also be viewed as the frequency slope shown in Figure 2.1.

Traditionally, the distance measurement of impulse radars that use Time-of-Flight (ToF) technologies is calculated as Equation 2.2.

$$R = \frac{c \cdot \Delta t}{2} \tag{2.2}$$

Where, R stands for range; Δt is the time duration between transmitted and received signals; $c \approx 3 \times 10^8$ is the light speed. Unlike traditional ToF radars that the precision of distance measurement depends on the accuracy of time measurement, FMCW radars take full advantage of the linear frequency profile. As shown in Figure 2.1, the time duration $\triangle t$ between the transmitted and received signals is transformed to the frequency difference $\triangle f$ for the distance calculation. The mathematical description of such transformation is formulated as Equation 2.3.

$$\triangle t = \frac{T_c \triangle f}{B} \tag{2.3}$$

This way, accurate distance measurement can be computed by simply observing the frequency difference between the transmitted and received signals.

Figure 2.1 also shows the existence of a constant phase shift f_D between the transmitted and received signals. This shift is known as Doppler frequency, which is caused by the moving objects. The calculation of the doppler frequency is conducted with the observations of f_D from consecutive chirps.

Modern FMCW radars are often designed with multiple transmitters and receivers. Using the phase angle differences between the signals received by different receivers, the angles of the reflected signals can also be calculated. Therefore, the information that can be acquired from FMCW radars consists of range (distance), doppler (velocity) and

angle.

2.1.1 3D FFT

Since the calculations of range, doppler and angle are based on the frequency or phase differences between the transmitted signals and received signals, Fast Fourier Transform (FFT) is considered as the primary method for the computations.

For range FFT, the overall process of the received signals from different objects is shown as Figure 2.2. Since the raw ADC signals from our FMCW radar are already

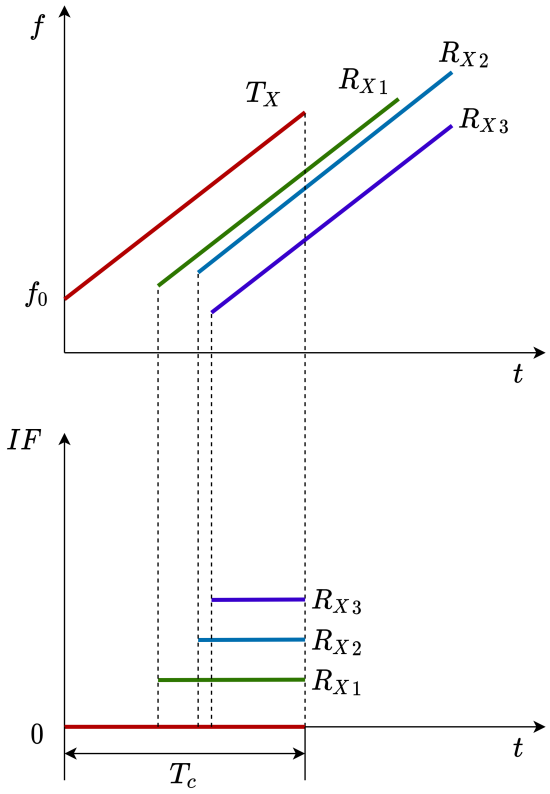

FIGURE 2.2: Top: FMCW signals from different receivers. Bottom: The frequency differences between the transmitted and received signals, also called Intermediate Frequencies (IF).

presented as the mixtures between the transmitted and received signals, the outputs of range FFT are directly interpreted as the frequency differences Δf, which is also known as **Intermediate Frequencies (IF)**. Combined with Equation 2.3, the distance between the object and the radar is thus formulated as Equation 2.4.

$$R = \frac{F_r \cdot T_c \cdot c}{2B} \tag{2.4}$$

Where, F_r is the IF; c is the speed of light. It is a fact that the frequency resolution of the mixed signal is bounded by the chirp frequency [70]. Therefore, IF F_r, which is the mixed

signal from the transmitted and received signals, is bounded by the chirp frequency $\frac{1}{T_c}$. Combined with Equation 2.4, a lower bound of range detections can be computed as $\frac{c}{2B}$. This lower bound is also called **range resolution**. It indicates the minimum range required for distinguishing two different objects in the same radar frame. On the other hand, since the input ADC signals of FFT are digitized, the frequency outputs are also discretized. According to the property of FFT, the output IF bandwidth is limited to the sampling frequency of the ADC signals. This IF bandwidth limitation also indicates radar's **maximum detection range**. Factors such as sampling rate and frequency slope are often pre-configured in order to constrain the maximum detection range.

For doppler FFT, calculations are done by accumulating a certain number of consecutive chirps due to several reasons. First, unlike range FFT that the computation is conducted with intermediate frequencies, the calculation of doppler FFT is based on the phase shifts between the transmitted and received signals, shown in Figure 2.1. Second, even by accumulating chirps, the total time duration is measured in the unit of μsec. Thus, the speed of the object can be viewed as a constant, which makes it feasible for the velocity calculations. Two major properties of doppler FFT, **doppler/velocity resolution** and **maximum detection velocity**, are determined by the total number of accumulated chirps and sweep time T_c respectively.

Angle FFT, unlike others, is determined by the on-board hardware setups of the radar. The number of transmitters and receivers directly decides the output quality of angle FFT. Based on the physical arrangement of transmitters and receivers, the received signals are usually arranged into an array, which is called **virtual antennas**. For most FMCW radars, the distance arrangements of transmitters and receivers are always proportional to the transmitted signal wavelength. Therefore, virtual antennas array is usually formed by concatenating all the received signals from different transmitters. This array is then processed by FFT and the output is interpreted as angles represented in polar coordinates. The process is also known as **Direction of Arrival (DOA)**. There are two types of angles that can be extracted from angle FFT, namely azimuth angle and elevation angle. Azimuth angle describes the horizontal position of the object and elevation angle indicates the vertical position. In our research, we only consider the azimuth angle due to the hardware limitation of the radar. Thus, in the following sections, we will call the angle FFT as **Azimuth FFT**.

2.1.2 Range-Azimuth-Doppler

As introduced above, different arrangements of the ADC signals are required for range FFT, doppler FFT and azimuth FFT. Therefore, it is reasonable to arrange the raw signals into the desired format. In practice, a preferable way is to formulate a 3D tensor with each dimension illustrating the number of digital signals within one chirp, number of chirps per frame and number of virtual antennas respectively. Figure 2.3 shows the pipeline of such data formatting. The formatted data are then directly fed into 3D FFT

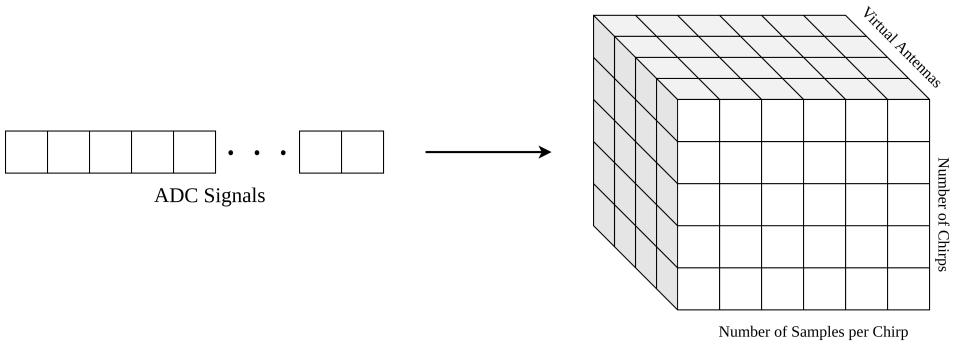

FIGURE 2.3: Data formatting for the raw ADC signals.

process for extracting the information on range, doppler and azimuth. The output is often called **Range-Azimuth-Doppler (RAD)**. As such format provides convenience for processing and interpreting radar data, it gradually becomes the main data format for both academic and industrial usages. Two sub-category formats, **Range-Azimuth (RA)** format and **Range-Doppler (RD)** format, can be extracted from RAD tensors by summating along the doppler axis and the azimuth axis respectively. Figure 2.4 shows the RA and RD representations. As RAD format is represented in a 3D tensor, it is always

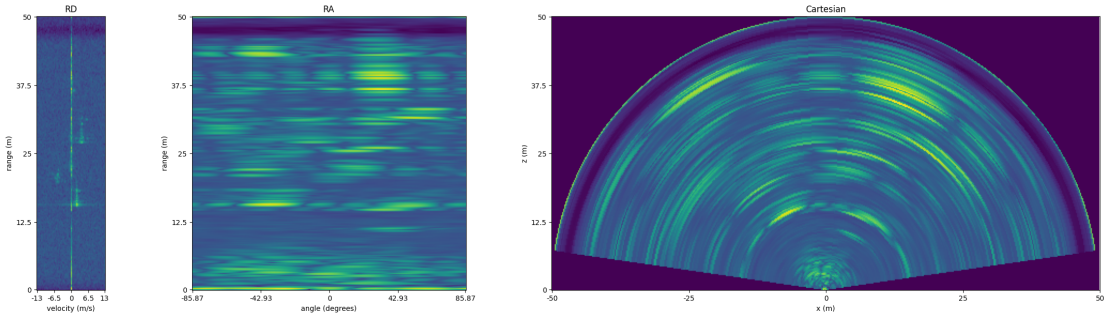

FIGURE 2.4: Left: Range-Doppler format. Middle: Range-Azimuth format. Right: Range-Azimuth representation in Cartesian coordinate

transformed into RA and RD formats for visualization. Similar to polar representation, RA format, shown in Figure 2.4, contains the information of range and angle. Thus, it is always transformed into Cartesian representation for higher readability.

A variance of signal processing algorithms on azimuth dimension exists as MUSIC [64]. This algorithm is able to provide angle detections with higher precision. However, due to the high computational power consumption, it is less popular than FFT in this field. In our research, we employ FFT as our radar DSP method in order to study the general performance of deep learning models on FMCW radar data.

2.1.3 Constant False Alarm Rate

Background noise is a common issue of FMCW radars. Various algorithms have been developed over the years to overcome this issue. Among them, Constant False Alarm Rate (CFAR) has gained certain popularity during recent years due to its robustness against various noise signals.

Figure 2.5 shows the data flow of CFAR algorithms. This type of algorithms uses

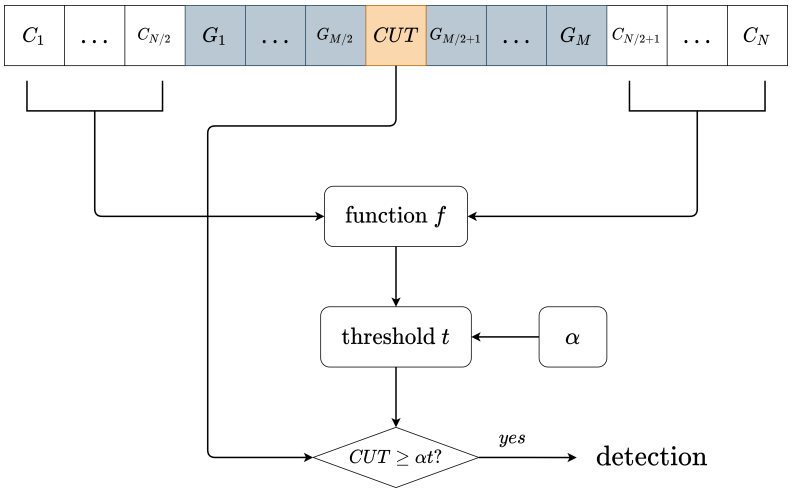

FIGURE 2.5: The illustration of CFAR algorithms in general. CUT is the Cell Under Test. Cells C_1 to C_N are for generating the CFAR threshold. G_1 to G_M are the guard cells.

sliding-window techniques for noise filtering. Each input cell from the RAD data is processed by this technique individually during the denoising process. The cell to be processed by the algorithm is often called Cell Under Test (CUT). At each CUT, a fixed-size window is defined with CUT centered, shown as all the cells in Figure 2.5. A set of guard cells is defined as the neighbouring cells of CUT, shown as G_1 to G_M in Figure 2.5. These cells are not considered for the computation during the process due to the reason that they usually belong to the same detection target as CUT. Cells that do not belong to either CUT or guard cells, shown as C_1 to C_N in Figure 2.5, are then passed into a predefined function f for computing a threshold. Afterwards, a tunable coefficient α is added for adjusting the threshold levels. The value of CUT is then evaluated against this threshold to define detections.

Based on different predefined functions, the implementations of CFAR algorithm vary from each other. Two CFAR sub-categories, namely Cell-Averaging CFAR (CA-CFAR) and Ordered-Statistic CFAR (OS-CFAR), have been widely employed to various applications. The function f of CA-CFAR is to take average over all input values. For OS-CFAR, inputs are sorted and the cell from a predefined location of the sorted input array is selected for defining the threshold. Algorithm 1 shows the detailed implementation of OS-CFAR algorithm. Practically, OS-CFAR provides higher performance but

much slower than CA-CFAR. Therefore, OS-CFAR is more preferable for academic usages while CA-CFAR has been seen more in the industry.

Algorithm 1: OS-CFAR Algorithm [81]

```
Initialize window cells size M + N;
Initialize guard cells size M;
Initialize the ordered statistic number j;
Set all cells to be detected as a set CUT_set;
for CUT ∈ CUT_set do
    Initialize all measurement cells C = {C_1, C_2, ..., C_N} w.r.t CUT;
    Sort cells C and extract the j − th element as C_j;
    if CUT ≥ α · C_j then
        The current CUT is marked as detection.
    else
        Continue for loop.
    end
end
```

2.2 Stereo Vision

Stereo vision, also known as binocular stereopsis, uses the perspective difference between two cameras for depth estimation. In our research, we use stereo vision along with Mask-RCNN [24] for instance-wise point-cloud generation. The following sections are separated into calibration, disparity estimation and triangulation to briefly introduce the general process of stereo depth estimation.

2.2.1 Intrinsic and Extrinsic Calibration

Modern cameras are developed based on the poin-hole camera techniques. The general process is to project the objects from 3D world coordinates onto a 2D image plane. For different cameras, different hardware setups normally lead to different Field of Views (FOVs) and distortion levels. **Camera intrinsic matrix** is defined to describe these properties. Equation 2.5 shows the mathematical model of real-world objects projected onto the image plane.

$$\begin{bmatrix} X_{image} \\ Y_{image} \\ Z_{image} \end{bmatrix} = \boldsymbol{P} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \boldsymbol{K}[\boldsymbol{I}|\boldsymbol{0}] \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} f & s & c_x \\ 0 & f & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \tag{2.5}$$

where, $[X_{image}, Y_{image}, Z_{image}]$ are the coordinates presented in the image plane and $[X, Y, Z]$ are the locations of objects in the world coordinates. $\boldsymbol{P}$ is the projection matrix from world coordinates to the image plane. The matrix $\boldsymbol{K}$ is the Camera Intrinsic Matrix. The formal mathematical description of the camera intrinsic matrix is shown as Equation 2.6.

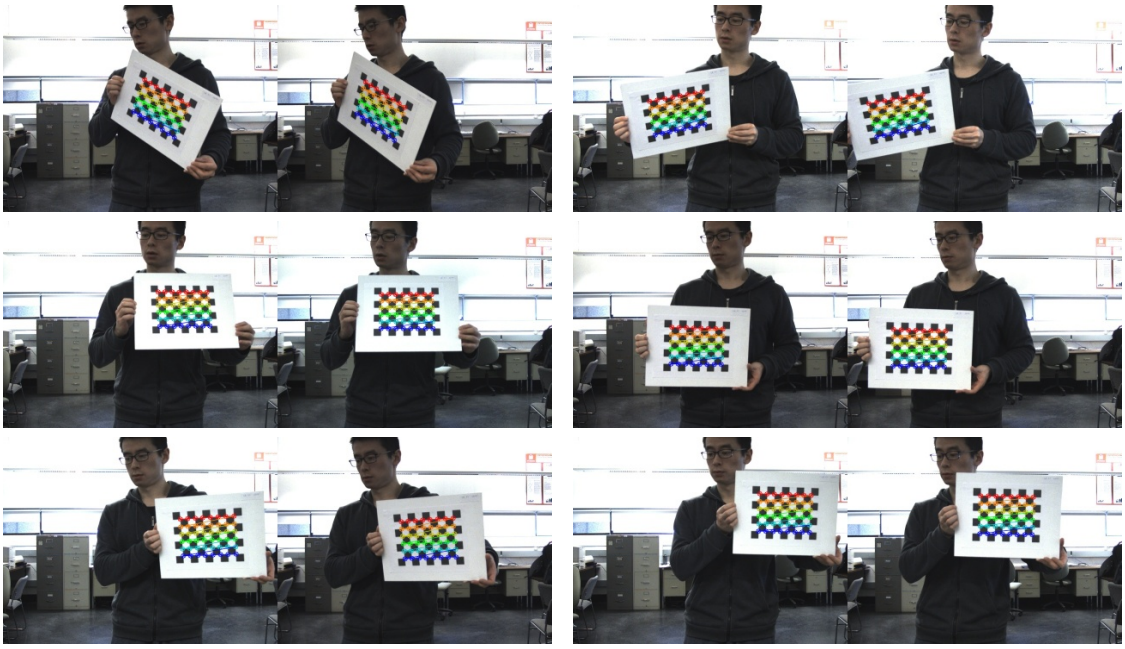

FIGURE 2.6: Examples of the chessboard image pairs taken from the stereo cameras for the calibration.

$$\boldsymbol{K} = \begin{bmatrix} f & s & c_x \\ 0 & f & c_y \\ 0 & 0 & 1 \end{bmatrix} \tag{2.6}$$

In stereo vision, in order to accurately estimate the distances between the sensors and the objects, the first step is to align the images taken from the two cameras. To do so, the projection matrices of both cameras $\boldsymbol{P_1}$ and $\boldsymbol{P_2}$ need to be computed. Equation 2.7 and 2.8 shows the definitions of those matrices.

$$\boldsymbol{P}_1 = \boldsymbol{K}_1[\boldsymbol{I} \;\; \boldsymbol{0}] = [\boldsymbol{I} \;\; \boldsymbol{0}] \tag{2.7}$$

$$\boldsymbol{P}_2 = \boldsymbol{K}_2[\boldsymbol{R} \;\; \boldsymbol{T}] = [\boldsymbol{R} \;\; \boldsymbol{T}] \tag{2.8}$$

Where $\boldsymbol{R}$ is called **Rotation Matrix** and $\boldsymbol{T}$ is **Translation Matrix**. The combination of these two matrices is called **Homogeneous Transformation Matrix** with the form shown in Equation 2.9.

$$\boldsymbol{H} = \begin{bmatrix} \boldsymbol{R}_{3\times3} & \boldsymbol{T}_{3\times1} \\ \boldsymbol{0} & 1 \end{bmatrix} = \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.9}$$

Since this matrix indicates the physical relationships between two image planes, it is often referred to as **Extrinsic Matrix** of the stereo cameras.

In our research, the calibration of the stereo cameras and the computations of their

intrinsic and extrinsic matrices are implemented with OpenCV[2]. Several sets of chessboard images are taken from both cameras during the calibration process, shown as Figure 2.6. Manual selections of the calibration images are also conducted to optimize the intrinsic and extrinsic matrices.

2.2.2 Disparity Estimation

After the derivations of the intrinsic and extrinsic matrices, the images taken from the left and right cameras are transformed onto the same 2D plane for the alignment. The process is also known as **Stereo Rectification**. As a result, the pixels of the same object on both images will be horizontally aligned. Due to the physical distance between the two cameras, a pixel gap can also be found between the projections of the same object on both images. This gap is known as **disparity**. Such disparity often indicates the distance between the sensor and the object. Therefore, a **disparity map** is preferable for illustrating the depth information of all the objects in the images. Figure 2.7 shows a sample of generated disparity maps.

FIGURE 2.7: A sample of generated disparity maps. Left: images taken from both stereo cameras. Right: disparity map. Darker the color, further the object.

One intensively studied area of the generation of disparity maps is stereo matching. The general understanding of stereo matching is to search for the pixel in the right image that most likely contains the same object as the target pixel in the left image. Typical window-based searching algorithms, such as line search and Global Block Matching algorithm, have been widely researched in recent years due to their high performance and fast processing speeds. In our research, we employed the implementation of Semi-Global Block Matching (SGBM) algorithm from OpenCV for the stereo matching. Equation 2.10 mathematically explains the SGBM algorithm [27].

$$C_{\boldsymbol{r}}^{A} = C(\boldsymbol{p}, d) + \min \begin{cases} C_{\boldsymbol{r}}^{A}(\boldsymbol{p} - \boldsymbol{r}, d) \\ C_{\boldsymbol{r}}^{A}(\boldsymbol{p} - \boldsymbol{r}, d-1) + P_1 \\ C_{\boldsymbol{r}}^{A}(\boldsymbol{p} - \boldsymbol{r}, d+1) + P_1 \\ \min_i C_{\boldsymbol{r}}^{A}(\boldsymbol{p} - \boldsymbol{r}, i) + P_2 \end{cases} \tag{2.10}$$

[2]https://opencv.org/

Where, $\boldsymbol{p}$ stands for target pixels in the left image; d stands for the maximum searching disparity channels; $\boldsymbol{r}$ is an array that consists of all the desired searching directions; P_1, P_2, P_3 are tunable parameters for confinements; C means cost function. For SGBM, the array $\boldsymbol{r}$ is always defined with 16 different directions, and the cost function C can be either sum of absolute difference or sum of square difference. Techniques, such as Weighted Least Square (WLS) filter and disparity normalization, are also applied after SGBM in order to improve the quality of disparity maps.

2.2.3 Triangulation

With the derived disparity maps and other information above, the pixel-wise depth estimation can be done by using triangulation geometry shown in Figure 2.8. The formal

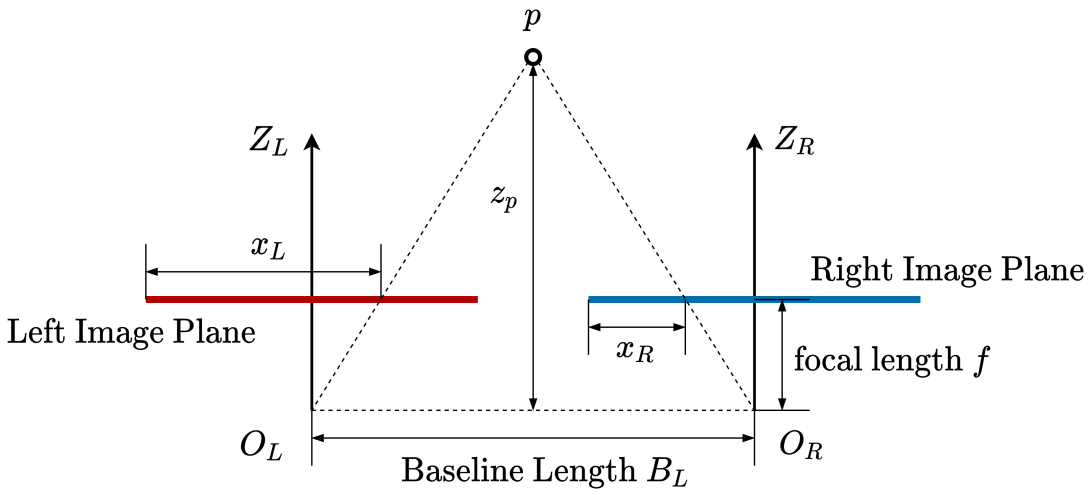

FIGURE 2.8: Triangulation geometry. x_L is the pixel location of point p on the left image; x_R is the location on the right image.

mathematical explanation is formed as Equation 2.11.

$$z_p = \frac{B_L \cdot f}{x_L - x_R} \tag{2.11}$$

Where, z_p stands for the depth of the target; B_L stands for the baseline distance between the two cameras; f means local length; $x_L - x_R$ is the disparity found by SGBM.

The maximum detection depth and the accuracy of depth estimation are determined by two major factors, namely image resolution and baseline length. The image resolution of the cameras used in this research is 1440×1080. However, such resolution was found problematic due to the reason that the noise levels of the generated disparity maps are considerably high, especially when objects are too far away from the sensors. To solve this issue, we experimented with different image resolutions along with different parameters for both SGBM algorithms and WLS filters. After several trials, we finally decided to use 800×600 as the input resolution. For baseline length, a large length

is always able to improve the accuracy of the detections for distant objects. Meanwhile, it also reduces the Field of View (FOV) of the stereo sensor. Since the maximum detection range of our radar is set to $50m$, the baseline length of the stereo is adjusted to fit the radar configurations in our research.

As the disparity maps are derived from the left and right images, image-based object detection algorithms [55, 41] and instance segmentation algorithms [24] can be employed for instance-wise depth estimation. In our search, we use Mask-RCNN [24] for instance-wise point-cloud generation. The generated point-clouds are then used for annotating the radar instances. Details are introduced in Chapter 4.

2.3 Deep Learning

Deep Learning (DL) is a thriving research topic in recent years. Its perspective is to use a **dataset** to learn a series of **parameters** that can perform a specific **task** without hard-coded rules or human understandings [63]. The formal mathematical description is as follow, given a dataset $\boldsymbol{X}$ that consists of inputs $\{x_1, x_2, ..., x_n\}$, a deep learning algorithm is defined as Equation 2.12 [63],

$$g(\boldsymbol{X}, \boldsymbol{\theta}) = \boldsymbol{W}_n \sigma_n(\boldsymbol{W}_{n-1} ... \sigma_2(\boldsymbol{W}_2 \sigma_1(\boldsymbol{W}_1 \boldsymbol{X}))), \quad \boldsymbol{\theta} = \boldsymbol{W}_1, \boldsymbol{W}_2, ..., \boldsymbol{W}_n \tag{2.12}$$

where, $\boldsymbol{\sigma} = \sigma_n, ..., \sigma_2, \sigma_1$ is a set of pre-defined functions called **activation functions**; $\boldsymbol{\theta}$ is a set of learnable parameters. The function g is also known as **model** in many research papers and applications.

The recent development of deep Learning has shown its incredible power in many research areas, such as Computer Vision (CV) and Natural Language Processing (NLP). In some specific tasks such as image recognition, it even exceeds human ability on the task performance.

It was not until the appearance of large datasets such as ImageNet [60] and COCO [39] that the deep learning development started to thrill. In ImageNet [60], Feifei Li *et al.*first revealed the super power of deep learning to the world. Since then, the development of deep learning models such as VGG [68], ResNet [25] and the development of task-targeted datasets such as KITTI [17] have attracted more and more researchers to dive into this field. Meanwhile, sequential data processing models such as Recurrent Neural Networks [86], Long Short Term Memory [28] and Transformer [76] have dramatically pushed the performance of deep learning on language processing.

This new trend of study is attributed to the development of several key DL components. First, the discovery of **backpropagation** enabled the gradient calculation in a computational efficient way. Second, optimization methods such as **Stochastic Gradient Descent (SGD)** [32] made the training achievable. The development of **activation functions** such as **rectified linear unit** further improved the models' ability to filter

noise signals. The discovery of **Multilayer Perceptron (MLP)** demonstrated the potential of very deep models and various **loss functions** have been introduced to target different tasks. Finally, with **Graphic Processing Unit (GPU) acceleration**, the fast training and inference speeds make the deployment of deep learning models feasible.

Based on different learning processes, deep learning is normally divided into **supervised learning**, **unsupervised learning** and **reinforcement learning**. Supervised learning is a learning process that relies on human inspections. Both ground truth labels and inputs are required in dataset $\boldsymbol{X}$ in order to enable the training. On the other hand, unsupervised learning is to learn the parameters $\boldsymbol{\theta}$ with only inputs from dataset $\boldsymbol{X}$. Reinforcement learning, unlike the other two, is to predict an action based on the given state in order to achieve the most rewarded outcome. This type of learning is usually seen in the tasks such as gaming, robotics and other decision-making related tasks.

Deep learning can also be categorized based on different types of tasks, such as Computer Vision (CV) and Natural Language Processing (NLP). In CV, a special type of deep learning model, **Convolutional Neural Networks (CNN)**, is widely used in various applications. The architectures such as VGG [68] and ResNet [25] mentioned above are all CNN-based models.

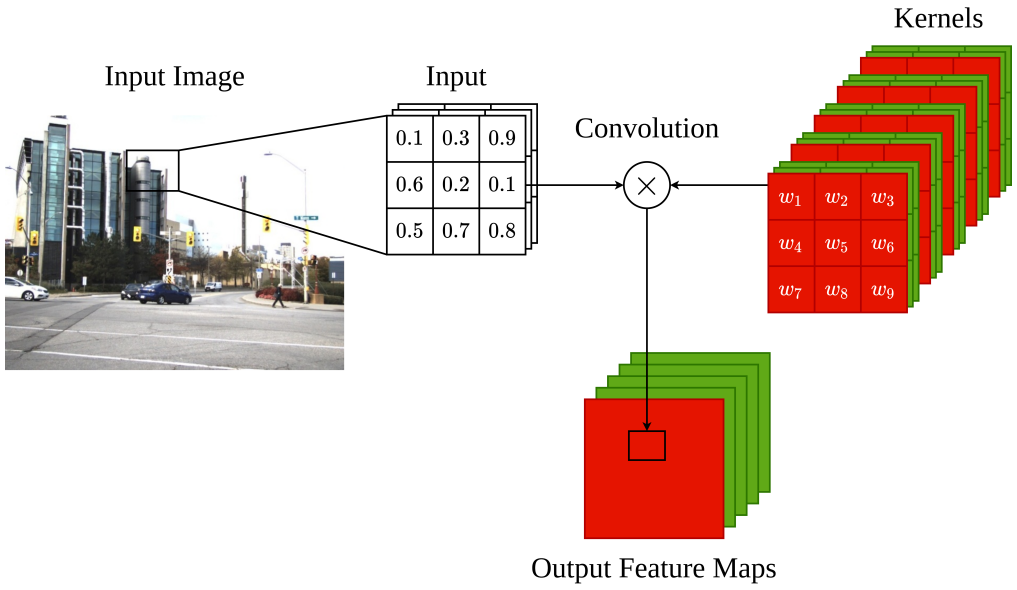

FIGURE 2.9: Data flow of convolutional layers.

Convolutional Neural Networks, also known as **CNNs**, are a specialized kind of neural networks for processing data that has a known grid-like topology [21]. There are two major components of CNN models, namely **convolutional layer** and **pooling layer**. Convolutional layer is developed under the inspiration of traditional kernel-based image processing algorithms. It defines a **kernel** and parameterizes all elements inside that kernel. The kernel is then slided through the entire input with convolution operation

described in Equation 2.13.

$$O(i,j) = (K * I)(i,j) = \sum_m \sum_n I(i+m, j+n) K(m,n) \tag{2.13}$$

Where, O is the output, which is also called **feature maps**; K is the kernel with (m,n) standing for a specific location inside it; I is the input with (i,j) illustrating the location inside the input. Figure 2.9 shows the data processing of convolutional layers. Hyper-parameters are defined to illustrate the behaviour of different types of convolutional layers. **Kernel size** defines the size of the shared parameters in each kernel; **Filters** shows the total number of output feature channels; **Stride** defines the output scale with respect to the input size; **Zero-padding** indicates the focus of the processing areas. Other types of convolutional layers such as **atrous convolutional layers** [7] are developed for the specific tasks. Pooling layer, as another basic component of CNNs, is a kernel-based operation without any parameterization. There are two popular types of pooling layers among computer vision researches, namely **max-pooling** layers and **mean-pooling** layers. Similar to the convolutional layer, pooling layer also defines a kernel for the operation. However, instead of filling the kernel with learnable parameters, it extracts the maximum or mean value from the kernel into the output feature maps. This operation can not only reduce the model size but also filter non-important features from the input.

Various tasks such as object detection, instance segmentation and semantic segmentation, have been explored with CNN-based models. The focus of our research is **object detection**. Since the dataset generation of our research is based on an **instance segmentation** algorithm, the background information of instance segmentation is also required for the process. In the following sections, the introductions of these two specific tasks are discussed.

2.3.1 Object Detection

In real world applications, the images often contain more than one object. Therefore, models that are designed for image recognition are not enough to identify and localize all the objects. As a result, a new task, object detection, is designed for targeting such problems.

Based on the applications, object detection algorithms are split into two categories: Generic object detection and Salient object detection [90]. Sub-categories of generic object detection are defined as face detection, pedestrian detection, and etc. As our research belongs to generic object detection, the characteristics of salient object detection are not discussed in this research.

Generic object detection is an instance-wise object-based task. It is usually classified into two sub-tasks based on different functionalities, namely **localization** and **classification**. The localization is realized through **bounding box regression**. Pixels that contain the information of the targeted object are included inside a bounding box and then used

for classification. The development of object detection algorithms is generally divided into two categories according to their architectures, namely **one-stage detectors** and **multi-stage detectors**.

R-CNN family [20, 19, 56, 24] is one of the keystones in object detection development. It is also the first two-stage detector. This type of model usually consists of two parts: a region proposal networks and a classification networks. The region proposal networks extract the regions that most likely contain the objects and feed them individually into the classification networks to identify those objects. The recent researches of multi-stage object detection have shown the highly achievable performance and the potential for more complicated tasks such as instance segmentation. However, the high computational consumption makes it difficult to apply those algorithms to edge devices and mobile devices.

One-stage detectors, on the other hand, are specifically designed for the high inference speed. They usually go through the input once and summarize all necessary information into the outputs. The interpretations of those outputs are then performed based on different models. YOLO [54] and SSD [41] are the two most famous one-stage detectors. Although they stand for two different detection methods, the mechanisms are similar. They both encode localization and classification information into one final output before decoding it. Figure 2.10 shows some examples of one-stage detectors. Recent development in this area has further improved both the runtime speeds and

FIGURE 2.10: Samples generated from YOLOv4 [2].

the precision of one-stage detectors , and the performance gap between one-stage and multi-stage detectors is gradually decreasing.

FIGURE 2.11: Samples generated from Mask-RCNN [24].

2.3.2 Instance Segmentation

Instance segmentation is also an instance-wise task but more complex than object detection. It can simultaneously predict the object class-label and the pixel-specific object instance-mask, and localize different classes of object instances present in various images [23]. Figure 2.11 shows some samples of instance segmentation.

Mask-RCNN, as a keystone in this field, first introduced the high-precision localization that can be achieved with instance segmentation. Since then, more researches such as YOLACT [3] and SOLO [80] have been conducted and results show that it has the potential for higher performance with much faster speed. However, due to the reason that it is still computationally expensive, the applications are still limited to certain devices.

In our research, Mask-RCNN [24] with implementations from Detectron2 [84] is applied to stereo images for instance-wise point-cloud generation. Details are introduced in Chapter 4.

Chapter 3

Related Work

The recent development of image-based deep learning algorithms has shown its dominant power in various tasks such as object detection, instance segmentation, super resolution and etc. Influenced by that, the development of autonomous radar is gradually turning into deep learning domain. However, its potential has not been researched to the full-extent due to various reasons. In this chapter, the recent radar studies are first introduced along with the problems they are facing. Then, the state-of-the-art object detection algorithms are introduced to provide an insight into the recent image-based deep learning development.

3.1 Deep Learning development of FMCW Radar

Rapid improvement of FMCW radar-based deep learning research has been witnessed during recent years. Tasks such as classification [51], object detection [11, 50, 43] and point-cloud segmentation [5] have been widely studied and experimented. One common issue of radar-based deep learning researches is the lack of proper public dataset. Thus, researchers tend to build their own datasets. Based on different types of radar inputs and outputs, these studies are split into different categories. In the following sections, we first introduce the techniques that have been used for generating radar datasets. Then, various researches are introduced based on different input formats.

3.1.1 Dataset

Unlike image-based deep learning development that public datasets such as COCO [39], ImageNet [60] and KITTI [17] can benchmark to various researches, the shortage of public radar datasets is caused by radar hardware variance and its highly configurable property. As introduced before, different FMCW chirp profiles such as frequency slops and digital sampling rates significantly impact the maximum detection range and range resolution of the radar. In addition, since the azimuth FFT is based on the total number of on-board transmitters and receivers, the angle resolution varies from radar to radar. As a result, various types of datasets have been built for different radar research purposes. On the other hand, issues like high noise level and low readability of radar

data make the dataset building process relatively difficult. In order to tackle this, other sensors such as Lidars and cameras are employed in various applications.

Camera, as the most intensively researched machine vision sensor, has been widely applied to radar researches [22, 5, 87, 36, 48] for building radar datasets. Brodeski *et al.* [5] use camera view for radar point-cloud segmentation. Lim *et al.* [36] employ camera Inverse Projection Mapping (IPM) for bird-eye-view transformation to match the radar sensor. Other researches [22, 87] transform the radar outputs onto camera frames for annotation. In addition, Nowruzi *et al.* [48] use monocular 3D reconstruction to build 3D maps for annotating radar data.

3D sensors such as stereo cameras and Lidar are also seen in radar researches for high-quality radar dataset generation. Palffy *et al.* [50] use a pair of stereo cameras for radar ground truth labelling. In [43, 11], Lidar is applied along with 3D object detection algorithms for generating decent ground truth boxes. Research [45] even uses multiple high-precision sensors such as Lidar, Synthetic Aperture Radar (SAR) and various cameras for annotating radar data.

Among all the researches above, different input data formats have been seen in different radar datasets. The focuses of those researches are thus split between these formats. Two major categories can be summarized from all the different radar input formats, namely cluster-based input format and spectrum-based input format. In the following sections, the radar researches are categorized and discussed based on these two types of formats.

3.1.2 Cluster-based Input Format

Cluster-based input format, as its name indicates, is the data format that contains a cluster of points with each point illustrating the location of the object. Figure 3.1 shows an example of this format. To generate such format, CFAR algorithm is first applied to the

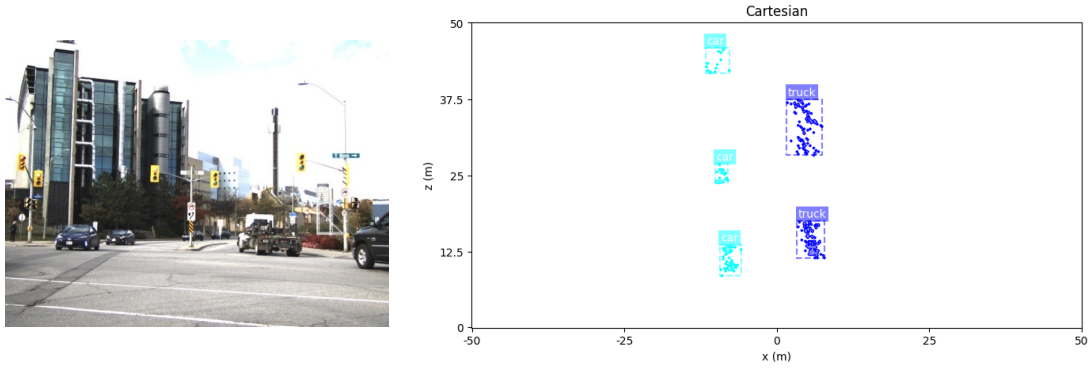

FIGURE 3.1: A sample of cluster-based input format.

radar data after 3D-FFT. Other traditional cluster-based processing algorithms such as K-means Clustering [1] and DBSCAN [15] are also required for the further noise filtering. For example, in [65], Schumann *et al.*use DBSCAN [15] to differentiate and extract

all the instances in the radar frames, then annotate each instance with the corresponding object in the camera frames. Afterwards, Long Short-Term Memory (LSTM) [28] and Random Forest [4] classifiers are compared for the classification task. Researches [62] and [61] , whose contributions are based on the work of [65], use the same data annotation algorithms but improve the performance of deep learning models furthermore.

One challenge of the cluster-based methods is that data could be mistakenly merged or separated due to the limited range resolution and the high noise level. This could not only increase the difficulties of the dataset generation but also potentially impact the performance of deep learning models. Details of the challenge are well stated in [50]. In order to solve this problem, the authors employ cropped spectrum-based data along with cluster-based data for radar object detection. As described in [50], this method effectively resolves the issues mentioned above.

Since the cluster-based input type is represented in a point-cloud manner, many Lidar-based deep learning algorithms are explored in the related cluster-based radar researches. Apart from LSTM and Random Forest mentioned above, Schumann *et al.* [66] employ multi-scale grouping module (MSG) with PointNet++ [52] for point-cloud based segmentation.

3.1.3 Spectrum-based Input Format

Spectrum-based input format, known as Range-Doppler (RD) spectrum, Range-Azimuth (RA) spectrum, Range-Azimuth-Doppler (RAD) spectrum and etc, keeps the consistency of the input data without any discretization. This will result in a performance boost for most deep learning algorithms. For example, Palffy *et al.* [50] successfully improve the performance of cluster-based algorithms by adding partial spectrum-based data. Meanwhile, it also increases the complexity of the input data, which could potentially impact the runtime speeds of those algorithms.

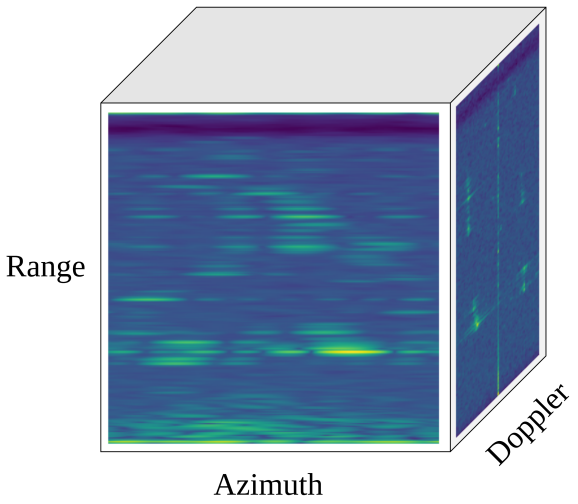

FIGURE 3.2: Range-Azimuth-Doppler (RAD) spectrum format.

In previous work, Brodeski *et al.* [5] take full-size information as Range-Doppler-Angles spectrum format from raw radar data for cluster-wise instance segmentation and pose estimation, where angels stand for a combination of azimuth and elevation angles. This full-size format requires high-resolution radar hardware that contains multiple transmitters and receivers aligned in both horizontal and vertical directions. To improve the generalization of spectrum-based radar research, Nowruzi *et al.* [48] and Major *et al.* [43] both decide to use Range-Azimuth-Doppler (RAD) spectrum as the input format for free space segmentation and vehicle detection. Figure 3.2 shows an example of this type of format.

2D formats such as Range-Doppler (RD) spectrum and Range-Azimuth (RA) spectrum are also widely researched in order to improve the runtime speed. Zhang *et al.* [87]

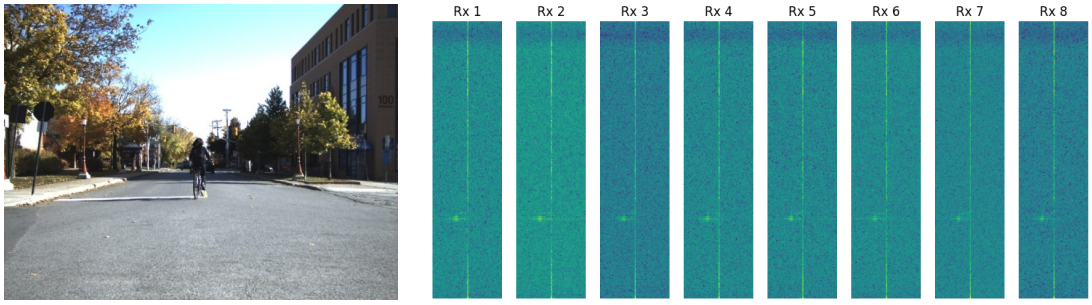

FIGURE 3.3: Samples of RD spectrum data received from 8 different virtual antennas.

use RD spectrum data received from each on-board receiver and rearrange them to form the input data for object detection. Figure 3.3 shows an example of this type of input. On the other hand, RA spectrum is a more popular input format due to the reason that it directly indicates the location of the object and can also be transformed into Cartesian coordinates for higher readability. Lim *et al.* [36] use RA spectrum as their input format for vehicle detection. Patel *et al.* [51] employ RA spectrum for classification task. As mentioned before, a variance of radar DSP on Azimuth dimension exists as MUSIC [64]. Therefore, RA spectrum format can also be classified into two sub-categories. The RA inputs of the above researches are computed by applying FFT. In [11], Dong *et al.*use the RA input format processed by MUSIC [64] for oriented box predictions. The difference between FFT and MUSIC is also discussed in [11]. Their results show that MUSIC provides higher quality outputs than FFT. However, as a computationally expensive algorithm, MUSIC also impacts the runtime speed of processing raw ADC signals. Therefore, in our research, FFT is selected as the major processing algorithm on Azimuth dimension.

One advantage of spectrum-based input format is the data continuity and coherence. Therefore, image-based deep learning algorithms such as convolutional layers are encouraged to apply to such input formats. For example, Patel *et al.* [51] use VGG [68] as its backbone architecture. Average-pooling layers are added after each convolutional

layer and the outputs are fed into a Multilayer Perceptron for the classification task. Brodeski *et al.* [5] build their high-level architecture based on a well-known two-stage networks, Faster-RCNN [56]. Inside their architecture, the region proposal networks "RD-Net" is developed with the same structure as U-Net[58]. FCN [67] and U-Net[58], two typical image-based deep learning architectures, are the most popular selections for recent radar researches [11, 87, 16, 48]. Apart from employing the state-of-the-art image-based models, some researchers build their architectures based on their own understandings of FMCW radars. In [43], the authors separate the RAD inputs along 3 axes and feed them into convolutional layers individually. The 3 types of outputs are then integrated and processed by additional layers for vehicle detections. Palffy *et al.* [50] directly use 3D convolution for processing input RAD data.

3.2 Image-based One-stage Object Detection

Modern one-stage object detection models [2, 37, 35, 13, 74] normally consist of 3 parts: a backbone, a neck and a detection head. Other techniques such as activation functions, regularization methods and data augmentation are researched and selected based on the specific tasks. Under the influence of the development in NLP [76], extra layers such as self-attention layers start to draw more attention. Researches [88, 49, 12, 53] show that self-attention layers can effectively boost the performance of deep learning models in Computer Vision domain. In this section, the classic and state-of-the-art backbones are first introduced. Neck layers are then briefly discussed before the introduction of anchor-based and anchor-free detection heads. Other techniques including batch normalization and self-attention layers are introduced at the end of this section.

3.2.1 Backbone

AlexNet [34], as one of the most influential backbones, was first brought into public in 2012. Since then, Convolutional Networks have been intensively studied with Computer Vision related tasks. Rectified Linear Unit (ReLU) and Max-pooling, which are also popularized from AlexNet, have also been brought to various applications. Under the inspiration of AlexNet, Simonyan *et al.* [68] pushed the performance of CNNs on image recognition much further with an architecture called VGG [68]. This backbone has been applied to various applications in later researches and it is still one of the most popular backbones till today. Table 3.1 shows the overall architecture of VGG [68]. Inception[71], invented after VGG, processes the inputs with multiple convolutional layers and max-pooling layers simultaneously before integrating them. As a result, considerably higher performance was achieved when compared to VGG. In addition, Inception also popularized the parallel processing of CNNs.

ResNet [25], another keystone backbone discovery, has pushed the performance of object detection algorithms towards the state-of-the-art level. Its simple yet reasonable

Blocks	**Structure**
Block 1	Conv2D(3, 64) + Conv2D(3, 64) + MaxPool(2, 2)
Block 2	Conv2D(3, 128) + Conv2D(3, 128) + MaxPool(2, 2)
Block 3	Conv2D(3, 256) + Conv2D(3, 256) + MaxPool(2, 2)
Block 4	Conv2D(3, 512) + Conv2D(3, 512) + MaxPool(2, 2)
Block 5	Conv2D(3, 512) + Conv2D(3, 512) + MaxPool(2, 2)
Fully Connected (FC) Block	FC(4096) + FC(4096) + FC(N_{class})
Activation	SoftMax

TABLE 3.1: The architecture of VGG [68].

block-based architecture has been tensely researched and deployed for both academic and industrial usages up till now. The original ResNet [25] consists of two types of blocks, **basic residual block** and **bottleneck residual block** shown in Figure 3.4. In each block, the input is convolved with sequential convolutional layers and the output feature maps are concatenated back with the input. The process not only makes the most of convolutional layers for feature extractions but also emphasizes the original input signals. This way, the feature details from the original input images can be reserved. Further improvements of ResNet-based researches such as CSPResNeXt [79] normally choose to keep the residual block structure but change the convolutional layers to other more advanced techniques.

MobileNet [29], as a hardware aware backbone, is designed for the deployment of deep learning models on edge devices and mobile devices. Its light-weight yet highly-efficient properties have encouraged the development of many applications on mobile devices.

Due to the recent popularity of Neural Architecture Search (NAS) [91], modern backbones such as EfficientNet [72] are designed by deep learning itself. As a direct result, the targeted backbones can not only outperform previously human designed backbones on the specific tasks but also provide more optimized runtime speed. For example, EfficientDet [73] outperforms most existing object detection models with both accuracy and speed by using EfficientNet [72] as the backbone.

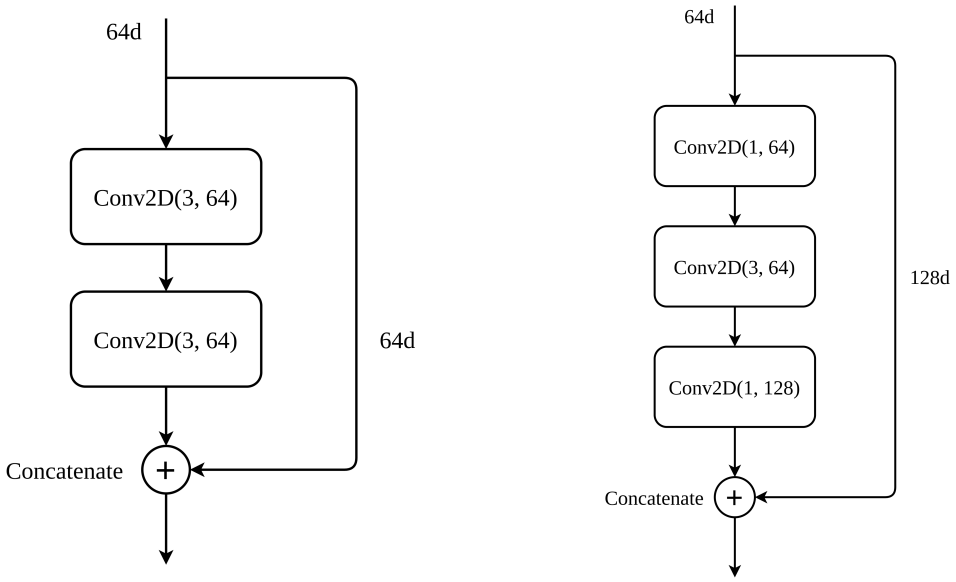

FIGURE 3.4: Residual blocks from ResNet [25]. Left: basic residual block. Right: bottleneck residual block. The numbers in "()" stand for kernel size and number of filters respectively.

3.2.2 Neck

The discovery of hierarchical-features based structure from RetinaNet [37] initiated a new trend of study in object detection, **neck layers**. Since then, an exponentially increasing number of related researches have been published in a very short period of time. Based on the functionalities, neck layers are generally split into two categories: additional blocks and path-aggregation blocks [2].

Additional blocks, as its name indicates, is to add additional layers on top of the output from the backbones. For example, Spatial Pyramid Pooling (SPP) [26] uses multiple pooling layers with different strides simultaneously on the output feature maps from the backbone. This way, it effectively relieves the problem that objects have various scales in the images due to perspective geometry. Atrous Spatial Pyramid Pooling (ASPP) [8] and Convolutional Block Attention Module (CBAM) [83] further improve the performance of SPP on various applications by using atrous pooling layers and self-attention layers respectively

Path-aggregation blocks take multiple feature maps from the backbones with different feature resolutions for the aggregation process. Feature Pyramid Networks (FPN) [38] and Path Aggregation Networks (PAN) [40], as two major keystones of this type, have been widely studied and employed in diverse tasks, such as object detection and instance segmentation. Further improvements such as NAS-FPN [18] and BiFPN [73] have also been studied with more advanced techniques such as NAS [91]. One advantage of path-aggregation blocks is that the multi-resolution output features can be fed into detection heads individually for improving the detections of small size objects.

3.2.3 Detection Head

Recent studies of one-stage detection head are generally split into two categories: anchor-based detection head and anchor-free detection head. Although anchor boxes have been considered as a crucial method for high precision detections, recent anchor-free development has shown an incredible performance improvement and the gap between these two methods is becoming less significant.

Starting from Region Proposal Networks (RPN) [56], the anchor-based detection head has shown its dominant power in the field of object detection. This type of detection head predicts the bounding boxes based on several predefined fixed-size boxes, which are also called **anchor boxes**. Normally, anchor boxes are defined by applying K-means Clustering [1] through the entire dataset. Therefore, they are able to present the overall box distributions of all the objects. During the process, the detection head finds the most suitable anchor boxes for the targeted objects and adjusts them with the correct scales. Figure 3.5 shows a sample of this method. Classic anchor-based detection

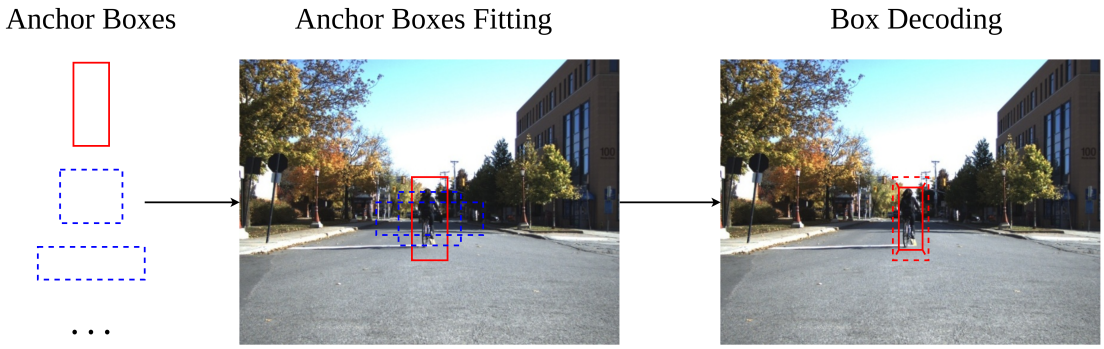

FIGURE 3.5: The general process of anchor-based one-stage detection heads.

heads such as SSD [41], YOLO [54] and RetinaNet [37], have been heavily researched recently and the various applications have shown their capability of being adapted into different tasks. Among them, YOLOv2 [6] defines the anchor boxes with fixed sizes, and introduces an "objectness" for identifying the existence of the objects. Equation 3.1 to Equation 3.4 show the box decoding process of YOLO [6].

$$x = \sigma(x_r) + c_x \tag{3.1}$$

$$y = \sigma(y_r) + c_y \tag{3.2}$$

$$w = p_w e^{w_r} \tag{3.3}$$

$$h = p_h e^{h_r} \tag{3.4}$$

where, σ means Sigmoid function; $[x, y, w, h]$ are box centers $(x,\ y)$ and sizes $(w,\ h)$; $[x_r, y_r, w_r, h_r]$ are raw box centers and sizes; $[c_x, c_y]$ are grid locations and $[p_w, p_h]$ are anchor box sizes. On the other hand, SSD [41] uses multi-resolution feature maps as the

inputs and thus outperforms YOLOv2 on the detections of small objects. In addition, it also introduces the aspect ratios as anchor boxes instead of fixed sizes. RetinaNet [37] further improves the structure of SSD with FPN neck layers. However, recent YOLO related studies, YOLOv3 [55] and YOLOv4 [2], show that with more advanced techniques such as neck layers and self-attention layers, both accuracy and speed can be significantly improved with much higher performance than SSD and RetinaNet.

The development of anchor-free detection heads, on the other hand, is still under intensive explorations. However, its increasing popularity has been seen in the most recent researches. Various methods have been explored during these years. For example, CornerNet [35] learns the positions of the top-left and bottom-right corners from the ground truth boxes, while CenterNet [13] predicts the bounding boxes as center points and offsets to the corners. FCOS [74], as a keystone in the anchor-free detection head development, interprets the output as a center and 4 offsets to the corresponding box boundaries. Zhang *et al.* [89] even show that with proper data mining, the performance of FCOS has no difference with anchor-based methods.

3.2.4 Choices of Operations

Other techniques including activations, regularizations, normalizations, and etc, have also been studied intensively. In our research, we call them **choices of operations**. Since the differences of them are involved in the specific implementations, we summarized a list of the currently popular operations for the choices in our experiments.

- **Activation Function(s):** ReLU, Leaky-ReLU, Mish [44], TanH, SoftMax, Sigmoid;
- **Regularization(s):** L1-regularizer, L2-regularizer, DropOut, Spatial DropOut [75];
- **Normalization(s):** BatchNormalization (BN) [30];
- **Loss Function(s) for Classification:** CrossEntropy (CE) [46], FocalLoss [37];
- **Loss Function(s) for Box Regression:** MSE, IoU, GIoU [57], Dice [69];

3.2.5 Extras: Self-attention Layers

Self-attention layers are developed based on the attention mechanism that consists of 3 parts: keys, queries and input values. However, instead of manually defining the keys and queries, self-attention layers parameterize them by using modern deep learning approaches. As a result, the output attention map can be learned automatically from the input features. Such a method has been widely applied to the NLP domain due to its ability of processing sequential data.

Since the success of Transformer [76] in NLP, researchers have been seeking the possibilities for self-attention layers bringing another revolution to the image-based models. The most recent research [12] even shows the Transformer itself can entirely substitute convolutional layers in the image recognition task. Other researches such as

[53] are also trying to develop a convolution-based self-attention method to substitute traditional convolutional layers in the tasks such as object detection and instance segmentation.

With the researches such as SAGAN [88] and SAUNet [49], different solutions for self-attention layers being deployed into traditional CNNs have been formed. In SAUnet [49], the authors propose a type of self-attention layers that uses two levels feature maps as the base to generate keys, queries and values. All the processes are done by convolutional layers. As a result, it shows a significant improvement in the semantic segmentation task. SAGAN [88], on the other hand, uses matrix multiplication in the proposed self-attention layers for generating the attention map. The results also show that this type of self-attention layers is able to improve the performance of image-based generative models.

In our research, both methods are applied to our models in order to study the general impact of self-attention layers on radar object detection.

Chapter 4

Dataset Generation

In our research, the radar dataset is built with an FMCW radar and a pair of stereo cameras. A novel auto-annotation algorithm is proposed for generating the ground truth labels. After that, manual corrections are conducted for further quality improvement. Our dataset is set as publicly available[1] for future research in this field. In this chapter, we first introduce the hardware setups and sensors registration. Then, the proposed auto-annotation algorithm is presented along with the instance extraction methods for both radar and stereo cameras. Finally, an overview of the radar dataset and the corresponding statistical analysis are introduced at the end of this chapter.

4.1 Hardware Setup and Sensor Registration

4.1.1 Hardware Setup

The sensors used in data collection consist of a *Texas Instrument* AWR1843-BOOST[2] and a pair of DFK 33UX273[3] stereo cameras from *The Imaging Source*. Both sensors are mounted onto a tripod as shown in Figure 4.1. The heights of the sensors are adjusted based on their Field of Views (FOVs).

As the radar is a highly configurable sensor, several experiments were conducted with different configurations in order to optimize the runtime speed and the capture frequency. Table 4.1 shows the radar and stereo configurations for the data capture in our research. Due to the limited resolution of elevation angle of the radar, we opt to only extract the information of range, doppler and azimuth angle. As introduced before, the raw ADC signals are normally formulated into 3D data with the dimensions indicating the number of samples within one chirp, the number of virtual antennas and the number of chips per frame. Therefore, the captured ADC data from our FMCW radar are sized $(256, 8, 64)$. For the stereo cameras, the baseline length of the two cameras, shown in Table 4.1, is defined with the considerations of both radar maximum detection range and the cameras' FOVs. Experiments show that such baseline length can lead to decent stereo depth estimation which ensures high-quality radar data annotations.

[1] http://www.site.uottawa.ca/research/viva/projects/raddet/index.html
[2] https://www.ti.com/
[3] https://www.theimagingsource.com/

FIGURE 4.1: Sensors setup: Stereo camera DFK 33UX273 is fixed at the top of the tripod; AWR1843Boost radar is installed in a yellow box which is attached to the middle of the tripod.

Some Parameters Setting	
Stereo Camera	
Parameter	Value
Image Resolution	1440×1080
Maximum Capturing Frequency	60 fps
Capturing Frequency	10 fps
Baseline Length	20 cm
Radar	
Parameter	Value
Maximum Range	50 m
Range Resolution	0.1953125 m
Maximum Velocity	13.4297698 m/s
Velocity Resolution	0.41968 m/s
Transmitters Enabled	2
Receivers Enabled	4
ADC Samples Per Chirp	256
Chirps Per Frame	64
Chirp Frequency Slope	30 $MHz/\mu sec$
Capturing Frequency	10 Hz

TABLE 4.1: Configurations of the sensors in this research.

The data synchronization is implemented with the Robotics Operating System (ROS). Timestamps are manually added to the received ADC data in order to synchronize with the cameras. During the data capture sessions, we also manually calibrated the recorded

timestamps from both sensors. As a result, the time difference between the two sensors is approximately $2ms$. We thus consider the data synchronization as durable for the next step.

The sensor registration in our research is implemented with a self-made trihedral corner reflector. As proven in many researches [10, 59], the detection rate of trihedral corner reflector is much higher than other objects due to its large Radar Cross Section (RCS), where RCS defines the object's ability to reflect the transmitted signals [14]. Figure 4.2 shows the setup of our calibration target. The colored styrofoam that attached

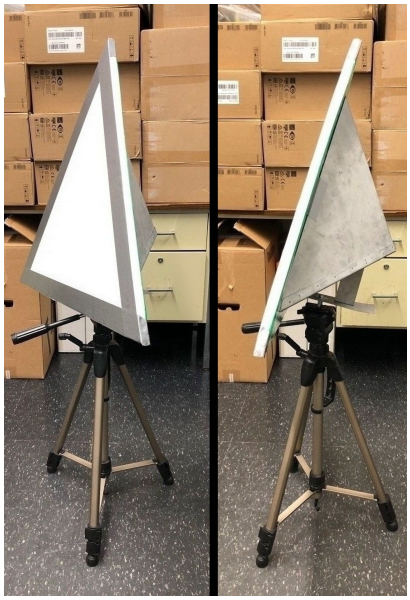

FIGURE 4.2: Calibration target: a triangle shape styrofoam attached to a trihedral corner reflector.

to the corner reflector is to ensure that the target can be easily extracted from the cameras by using traditional image process techniques such as color thresholding. Since styrofoam is mostly made out of the air, its impact on the radar can be neglected.

4.1.2 Sensors Registration

The goal of sensors registration is to find a projection matrix that can project the detections from one sensor frame to another. This projection matrix normally contains a Homogeneous Transformation Matrix. The mathematical description of the Homogeneous Transformation Matrix is shown as Equation 4.1.

$$\boldsymbol{H} = \begin{bmatrix} \boldsymbol{R} & \boldsymbol{T} \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.1}$$

Where, $\boldsymbol{H}$ is the Homogeneous Transformation Matrix; $\boldsymbol{R}$ is called rotation matrix and $\boldsymbol{T}$ as translation matrix. This Homogeneous Transformation Matrix describes the geometry relationships between two different frames in a form of 6 degrees of freedom.

In our research, since the elevation angle is not considered, the positioning information derived from the radar is presented in the 2D birds-eye-view format. On the other hand, the output of the stereo cameras is normally formulated in the 3D point-cloud format. Therefore, the projection geometry in our research is set from the stereo frame to the radar frame. Similar setups have been seen in [47]. Inspired by this research, we formulated the geometry relationship between the stereo cameras and the radar as Equation 4.2.

$$\boldsymbol{h}^{radar}(\boldsymbol{x}_{world}) = \boldsymbol{P}_{3D\to 2D}\boldsymbol{H}^{radar}_{stereo}\boldsymbol{h}^{stereo}(\boldsymbol{x}_{world}) \tag{4.2}$$

Where, $\boldsymbol{x}_{world}$ means the coordinates of the object in the world frame; $\boldsymbol{h}^{radar}$ stands for the location of the object in the radar frame; $\boldsymbol{h}^{stereo}$ indicates the location of the object in the stereo frame; $\boldsymbol{P}_{3D\to 2D}$ is the projection matrix from a 3D point (x, y, z) to a 2D point (x, z); $\boldsymbol{H}^{radar}_{stereo}$ is the homogeneous transformation matrix from the stereo frame to the radar frame. Combined with Equation 4.1, Equation 4.2 can also be expanded as Equation 4.3.

$$h^{radar}(\boldsymbol{x}_{world}) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot h^{stereo}(\boldsymbol{x}_{world}) \tag{4.3}$$

$$h^{radar}(\boldsymbol{x}_{world}) = \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{31} & r_{32} & r_{33} & t_3 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot h^{stereo}(\boldsymbol{x}_{world}) \tag{4.4}$$

Therefore, the final form of our projection matrix $\boldsymbol{P}^{radar}_{stereo}$ is formulated as Equation 4.5

$$\boldsymbol{P}^{radar}_{stereo} = \boldsymbol{P}_{3D\to 2D}\boldsymbol{H}^{radar}_{stereo} = \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{31} & r_{32} & r_{33} & t_3 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.5}$$

Since this projection matrix contains 8 parameters. a minimum number of 8 calibration locations need to be set up in order to conduct the sensors registration.

In real world applications, signals received from different sensors always contain noises, which are normally considered as zero-mean white noises. Therefore, the sensor measurements $\boldsymbol{h}^{radar}$ and $\boldsymbol{h}^{stereo}$ in our research are assumed as Equation 4.6 and Equation 4.7.

$$\boldsymbol{h}^{radar}(\boldsymbol{x}_{world}) = \boldsymbol{x}_{radar} + \boldsymbol{\omega}_{radar}, \quad \boldsymbol{\omega}_{radar} \sim \mathcal{N}(0, \boldsymbol{Q}_{radar}) \tag{4.6}$$

$$\boldsymbol{h}^{stereo}(\boldsymbol{x}_{world}) = \boldsymbol{x}_{stereo} + \boldsymbol{\omega}_{stereo}, \quad \boldsymbol{\omega}_{stereo} \sim \mathcal{N}(0, \boldsymbol{Q}_{stereo}) \tag{4.7}$$

Where, $\boldsymbol{x}_{radar} = [x_{radar}, z_{radar}]^T$ is the ground truth position of the object in the radar frame; $\boldsymbol{x}_{stereo} = [x_{stereo}, y_{stereo}, z_{stereo}]^T$ is the true position of the object in the stereo frame; $\boldsymbol{x}_{world} = [x_{world}, y_{world}, z_{world}]^T$ indicates the location of the object in the world frame; $\boldsymbol{\omega}_{radar}$ and $\boldsymbol{\omega}_{stereo}$ are the noise distributions of both sensors. The descriptions of covariance matrices $\boldsymbol{Q}_{stereo} \in \mathbb{R}^3$ and $\boldsymbol{Q}_{radar} \in \mathbb{R}^2$ are shown as Equation 4.8 and Equation 4.9.

$$\boldsymbol{Q}_{radar} = \begin{bmatrix} \sigma^2_{x_{radar}} & \sigma_{x_{radar}}\sigma_{z_{radar}} \\ \sigma_{x_{radar}}\sigma_{z_{radar}} & \sigma^2_{z_{radar}} \end{bmatrix} \tag{4.8}$$

$$\boldsymbol{Q}_{stereo} = \begin{bmatrix} \sigma^2_{x_{stereo}} & \sigma_{x_{stereo}}\sigma_{y_{stereo}} & \sigma_{x_{stereo}}\sigma_{z_{stereo}} \\ \sigma_{x_{stereo}}\sigma_{y_{stereo}} & \sigma^2_{y_{stereo}} & \sigma_{y_{stereo}}\sigma_{z_{stereo}} \\ \sigma_{x_{stereo}}\sigma_{z_{stereo}} & \sigma_{y_{stereo}}\sigma_{z_{stereo}} & \sigma^2_{z_{stereo}} \end{bmatrix} \tag{4.9}$$

Where, σ means standard deviation.

In our research, the noise distributions of the sensors are computed with Gaussian regressions. The implementation details are described as follow: For each sensor at each calibration location, multiple frames of the calibration target are captured. The positioning detections of those frames are then concatenated and processed by Gaussian regression. The output mean value and covariance matrix are considered as the true position of the calibration target and the sensor's noise distribution.

With the derivations of the sensor measurements and noise distributions, Equation 4.2 is then re-written to Equation 4.10.

$$\boldsymbol{x}_{radar} + \boldsymbol{\omega}_{radar} = \boldsymbol{P}_{3D\to2D}\boldsymbol{H}^{radar}_{stereo}\left(\boldsymbol{x}_{stereo} + \boldsymbol{\omega}_{stereo}\right) \tag{4.10}$$

$$\left(\boldsymbol{x}_{radar} - \boldsymbol{P}_{3D\to2D}\boldsymbol{H}^{radar}_{stereo} \cdot \boldsymbol{x}_{stereo}\right) + \left(\boldsymbol{\omega}_{radar} - \boldsymbol{P}_{3D\to2D}\boldsymbol{H}^{radar}_{stereo} \cdot \boldsymbol{\omega}_{stereo}\right) = 0 \tag{4.11}$$

$$\left(\boldsymbol{x}_{radar} - \boldsymbol{P}^{radar}_{stereo} \cdot \boldsymbol{x}_{stereo}\right) + \left(\boldsymbol{\omega}_{radar} - \boldsymbol{P}^{radar}_{stereo} \cdot \boldsymbol{\omega}_{stereo}\right) = 0 \tag{4.12}$$

As introduced before, minimum 8 calibration locations are required for the sensors registration. Therefore, the points $\boldsymbol{x}_{radar}, \boldsymbol{x}_{stereo}$ and the noises $\boldsymbol{\omega}_{radar}, \boldsymbol{\omega}_{stereo}$ in Equation 4.12 are substituted with the sets $\boldsymbol{X}_{radar}$, $\boldsymbol{X}_{stereo}$, $\boldsymbol{\Omega}_{radar}$ and $\boldsymbol{\Omega}_{stereo}$.

$$\boldsymbol{X}_{radar} = \{\boldsymbol{x}_{radar1}, \boldsymbol{x}_{radar2}, \boldsymbol{x}_{radar3}, ...\} \tag{4.13}$$

$$\boldsymbol{X}_{stereo} = \{\boldsymbol{x}_{stereo1}, \boldsymbol{x}_{stereo2}, \boldsymbol{x}_{stereo3}, ...\} \tag{4.14}$$

$$\boldsymbol{\Omega}_{radar} = \{\boldsymbol{\omega}_{radar1}, \boldsymbol{\omega}_{radar1}, \boldsymbol{\omega}_{radar1}, ...\} \tag{4.15}$$

$$\boldsymbol{\Omega}_{stereo} = \{\boldsymbol{\omega}_{stereo1}, \boldsymbol{\omega}_{stereo1}, \boldsymbol{\omega}_{stereo1}, ...\} \tag{4.16}$$

In our research, in order to improve the accuracy of the projection matrix $\boldsymbol{P}^{radar}_{stereo}$, more than 8 calibration locations are defined during the calibration process. Therefore, instead of direct derivation, an optimization method is preferred for the calculation of the projection matrix. As a result, the derivation of the projection matrix $\boldsymbol{P}^{radar}_{stereo}$ in our

research is defined as Equation 4.17.

$$\boldsymbol{P}_{stereo}^{radar} = \arg\min_{\boldsymbol{P}_{stereo}^{radar}} \left[(\boldsymbol{X}_{radar} - \boldsymbol{P}_{stereo}^{radar} \cdot \boldsymbol{X}_{stereo}) + (\boldsymbol{\Omega}_{radar} - \boldsymbol{P}_{stereo}^{radar} \cdot \boldsymbol{\Omega}_{stereo}) \right]^2 \tag{4.17}$$

To implement Equation 4.17, we first chose a parking spot as the experiment area. 10 random calibration locations were selected during the registration process. At each location, consecutive 50 frames from each sensor were taken for the noise calculations. The final projection matrix was computed based on the equations introduced above. Figure 4.3 shows results of the derived projection matrix.

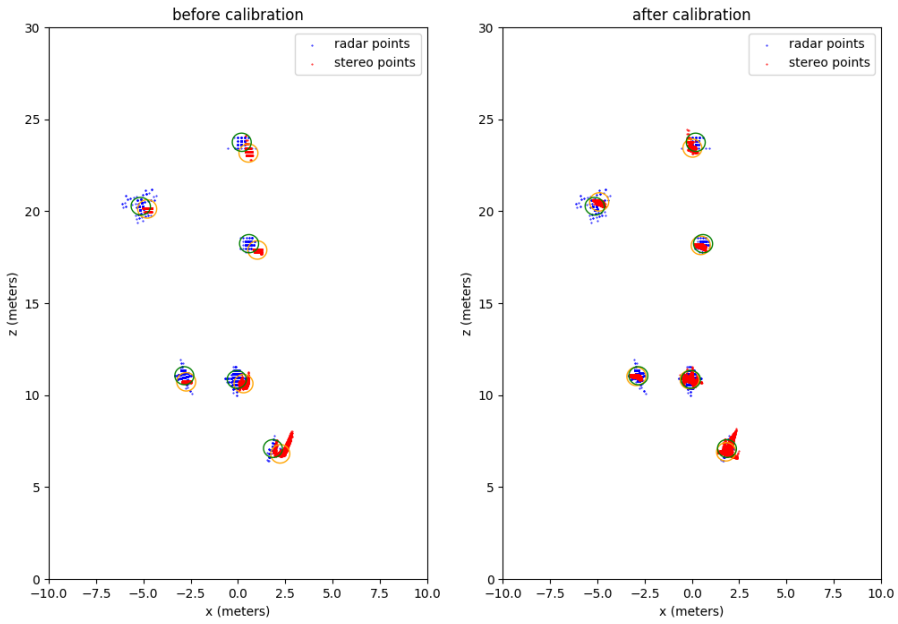

FIGURE 4.3: Samples of sensors registration. Left: points before calibration. Right: points after calibration. Colored circles describe the covariance of the noise distributions at each calibration location.

As shown in Figure 4.3, not only the centers of the projected stereo clusters overlap with the centers of the radar clusters, but also the covariances of their distributions become much similar. This further indicates that the derived projection matrix not only precisely projects the points from the stereo frame to the radar frame but also increases the similarities of their noise distributions. However, it also shows that the accuracy starts to decrease when the object gets too far away from the sensors. This is due to the reason that the noise distributions of the two sensors change when the distance between the object and the sensors changes , and the changing rates are not the same. For example, assume the noise level of the radar changes linearly with the distance to the object. The noise level of the stereo cameras may not follow the same trend. On the other hand, Equation 4.17 is derived under the assumption of the sensor noise levels being Gaussian distributions. This may lead to the false derivation of the true noise distributions, since the sensors may not behave ideally under various conditions.

Although the errors can be observed from Figure 4.3, the results still show an overall high accuracy. Thus, we consider the errors as neglectable and the derived projection matrix as durable for the data capture and annotation.

4.2 Data Auto-annotation

Our proposed auto-annotation algorithm is developed on top of traditional radar DSP. It consists of 3 processes: radar instances extraction, stereo instance-wise depth estimation and auto labelling. The details are introduced in the following sections.

4.2.1 Radar Data Pre-processing

The radar pre-processing in our research is conducted based on traditional radar DSP. First, 3D-FFT is performed on the received ADC data. The output RAD tensors are sized $(256, 256, 64)$ as zero-paddings are applied during the Azimuth FFT. 2D OS-CFAR, as a robust de-noise algorithm, is then applied to the Range-Doppler dimensions of the RAD tensors for radar instance extractions. The parameters of 2D OS-CFAR are fine tuned with multiple experiments, shown in Table 4.2. The entire fine tuning process is

2D OS-CFAR	
Parameter	Value
Window Size on Range-Doppler	$(12, 6)$
Guard Cell Size on Range-Doppler	$(4, 2)$
Order statistic	0.65
Parameter α	3

TABLE 4.2: Parameter settings of 2D OS-CFAR used in our research.

conducted under the considerations of both accuracy and processing speed. The direct output from 2D OS-CFAR is a detection mask with the detected points marked as 1 and background points as 0 on the Range-Doppler spectrum. A sample is shown as the right images in Figure 4.4. Peak values on Azimuth dimension are then used for finding the detections. Due to the reason that the detections of dynamic objects on the doppler axis are normally richer than the static objects, we thus opt to only consider dynamic objects in our research. As the radar is set stationary, the detections of absolutely stationary objects are manually filtered out. Only absolutely dynamic objects are considered in the following experiments.

In order for us to visually inspect the quality of traditional radar DSP, experiments were conducted with this pre-processing method. After large observations, several characteristics of the objects on the RD spectrum were found and summarized. First, the performance of 2D OS-CFAR is unstable. It sometimes ignores the objects with weak reflected signals, and sometimes false detects the noise signals as the objects. Second, rigid bodies such as cars and trucks, can be easily detected and the patterns of them

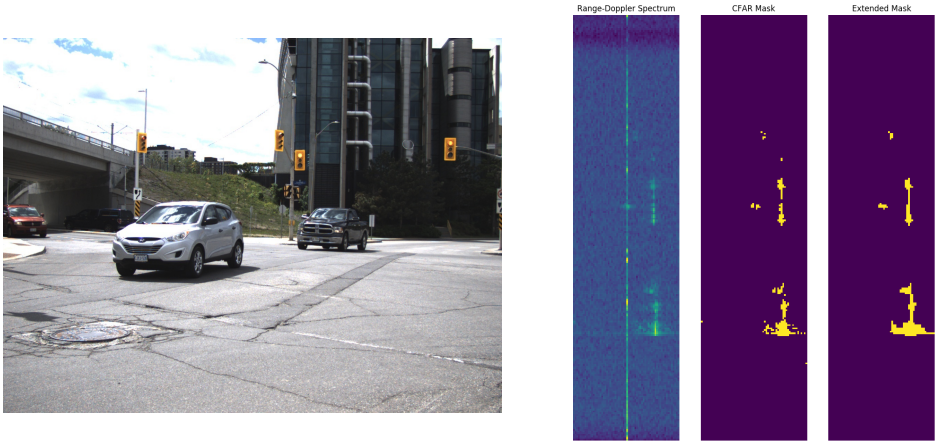

FIGURE 4.4: Left: RGB image for visualization; right: 3 images with order: range-doppler(RD) spectrum, detection mask from 2D OS-CFAR, extended mask from morphological extension.

shown on the RD spectrum always appear as linear patterns. Third, due to the limited range resolution shown in Table 4.1, the complex motions of the human bodies can hardly be observed in the radar frames. Thus, objects such as pedestrians and bicycles can also be considered as rigid bodies in our research. Under such assumptions, all the detected patterns on the RD spectrum can be considered as linear patterns. Therefore, the discrete patterns shown in Figure 4.4 can be linearly connected to enrich the detections.

Figure 4.4 also shows a sample of linear extension conducted on a RD mask derived from 2D OS-CFAR. The process can be easily done with an image processing algorithm called **morphological extension**. In our research, the implementation of morphological extension from OpenCV is employed for the process. By doing this, not only the detections on the RD dimensions are enriched, but also the small noisy detections are filtered out. In order to verify our assumptions, Azimuth peak is conducted on both extended and non-extended masks. The output RA maps are then transformed into Cartesian coordinates for higher readability. Figure 4.5 shows the final detections of both methods. The enrichment of the detections on the Cartesian maps proves that such method

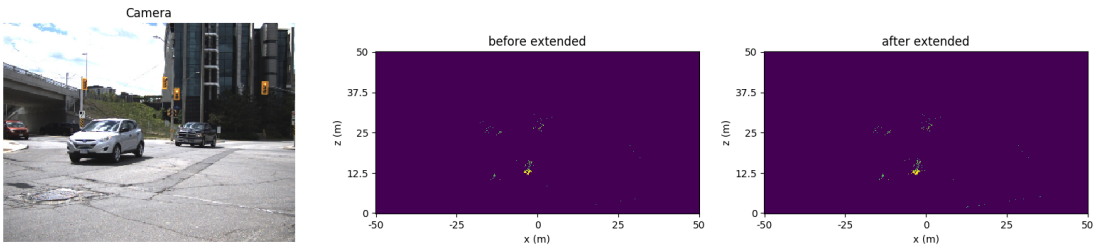

FIGURE 4.5: Left: RGB image for visualization; middle: birds-eye-view detections from non-extended RD mask; right: birds-eye-view detections from extended RD mask.

is durable for radar instance extractions. Meanwhile, there are still noise points being added during the morphological extension process. In order to further improve the detection quality and to distinguish the radar instances, 3D DBSCAN [15] is added after the process of Azimuth peak. With certain fine-tuned parameters, high-quality radar instances are extracted from the input RAD tensors.

This proposed radar data pre-processing method is considered as an improvement of traditional radar DSP. It not only enrichs the detection rate of dynamic objects, but also filters the noise signals to a certain extent. The whole data pipeline is concluded as Figure 4.6. The high-quality radar instances are the base in this research for generating

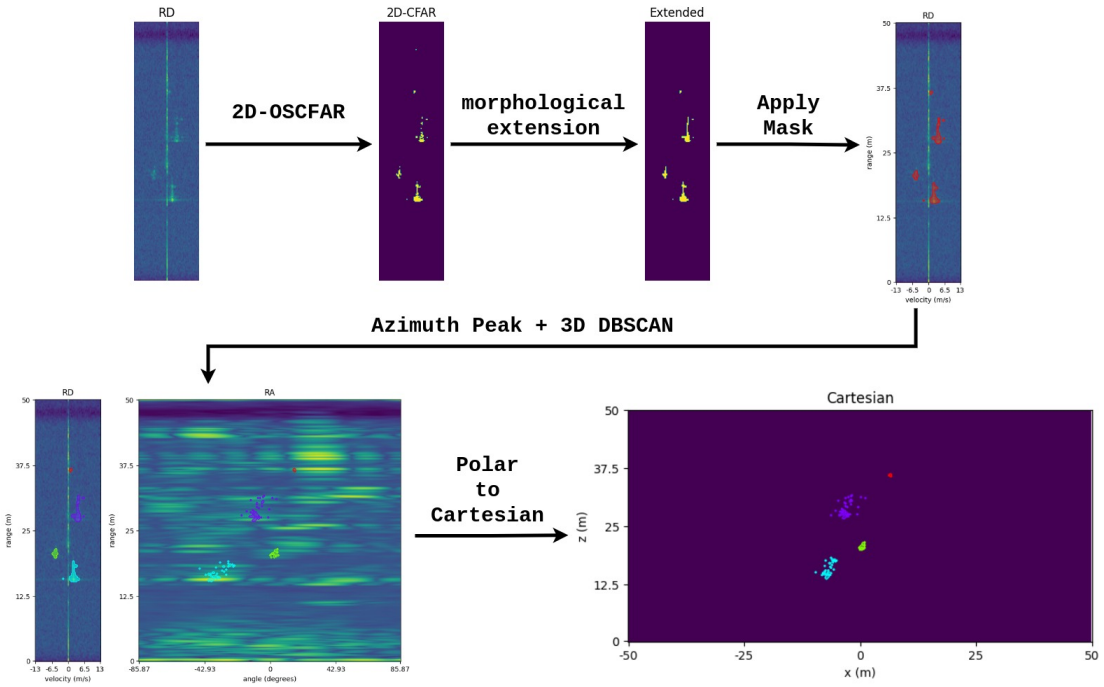

FIGURE 4.6: The data pipeline of the proposed radar data pre-processing method.

accurate bounding boxes.

In order to generate the radar dataset, all the extracted radar instances need to be categorized. Due to the low-readable properties of the rada data themselves, other sensors such as Lidars and cameras are needed for the categorization. In our research, the stereo cameras are used for this purpose. Details are introduced below.

4.2.2 Stereo Depth Estimation

The stereo depth estimation in our research is implemented with OpenCV. Both cameras were calibrated with 30 sets of chessboard images. During the data capture, the captured frames were rectified and processed by SGBM for generating disparity maps. A sample of the disparity maps is shown as the right image in Figure 4.7. As the im-

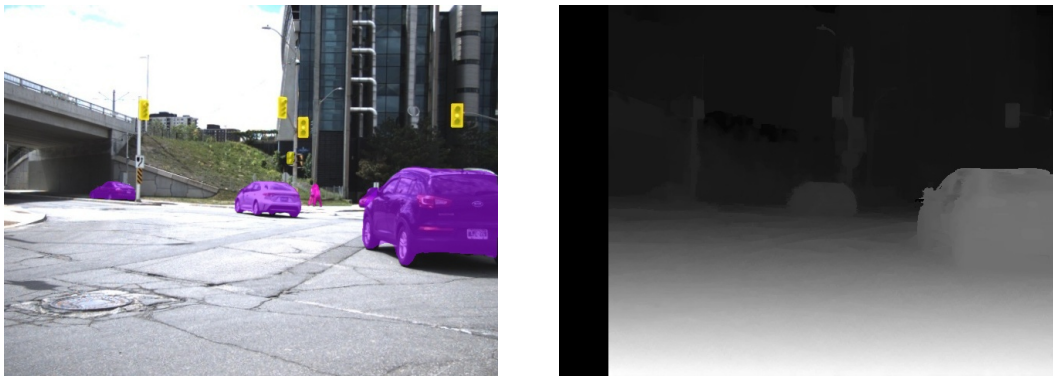

FIGURE 4.7: Left: instance segmentation from Mask-RCNN [24]. Right: disparity map from stereo vision.

plementation of SGBM from OpenCV is conducted based on the left image, the pixel locations of the generated disparity maps are the same as the left images. Based on this property, Mask-RCNN [24] is applied to the left images for the instance-wise segmentation, shown as the left image in Figure 4.7. The implementation and the pre-trained weights of Mask-RCNN are downloaded from Detectron2 [84]. The generated masks are then applied to the disparity maps along with the category predictions. Combined with the usage of triangulation, instance-wise point clouds along with the predicted categories are then generated. We call these instances stereo instances.

For each stereo instance, the 3D points are projected onto the radar frame by using previously derived projection matrix. This way, both locations and categories of the objects can be transformed onto the radar frame. Figure 4.8 shows a sample of this process.

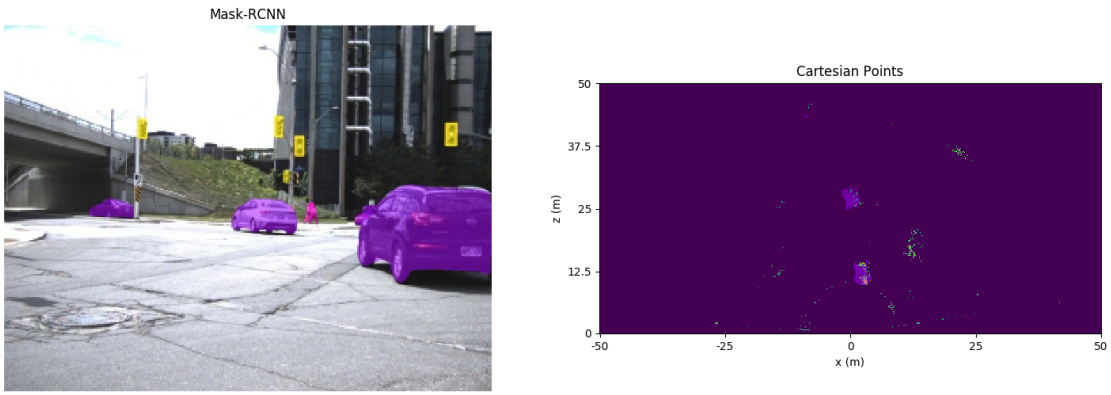

FIGURE 4.8: A sample of stereo instances projected onto the radar frame. The colored points are the projections of the stereo points with the colors illustrating the categories. The green points are the radar detection points.

Our proposed auto-annotation algorithm is developed based on a combination of the radar instances and the stereo instances described above. Details are introduced

below.

4.2.3 Auto-annotation algorithm

Our proposed auto-annotation algorithm is a center-based matching algorithm. With the radar instances I_{radar}, the stereo instances I_{stereo} and a predefined hyper-parameter $D_{threshold}$ which stands for the distance threshold for the matching process, the auto-annotation algorithm is described as Algorithm 2. Where, `norm` stands for Euclidean

Algorithm 2: Auto-annotation algorithm

Input: I_{stereo}, I_{radar}, $D_{threshold}$, $\boldsymbol{P}_{stereo}^{radar}$
Output: $Annotation_{radar}$
Initialize $Annotation_{radar}$
for $I_s \in I_{stereo}$ **do**
  Initialize D_{min}, I_{target}
  $C_{stereo} = \boldsymbol{P}_{stereo}^{radar} \times \texttt{findCenter}(I_s[\text{“}cluster\text{”}])$
  for $I_r \in I_{radar}$ **do**
    $C_{radar} = \texttt{findCenter}(I_r[\text{“}cluster\text{”}])$
    $d = \texttt{norm}(C_{stereo} - C_{radar})$
    if $d \leq D_{min}$ **then**
      $D_{min} = d$
      $I_{target} = I_r$
  if $D_{min} \leq D_{threshold}$ **then**
    $class = I_s[\text{“}class\text{”}]$
    $box_{3D} = \texttt{getBox}(I_{target}[\text{“}RADmask\text{”}])$
    $box_{2D} = \texttt{getBox}(I_{target}[\text{“}cluster\text{”}])$
    $Annotation_{radar} \leftarrow \{class, box_{3D}, box_{2D}\}$
    $I_{radar}.\texttt{remove}(I_{target})$

distance; `getBox` is to extract the bounding box from either the input point cloud or the input mask; `findCenter` is to find the centers of the given clusters using Gaussian regressions; $\boldsymbol{P}_{stereo}^{radar}$ is the projection matrix obtained in sensors registration. Figure 4.9 shows a result from our auto-annotation algorithm.

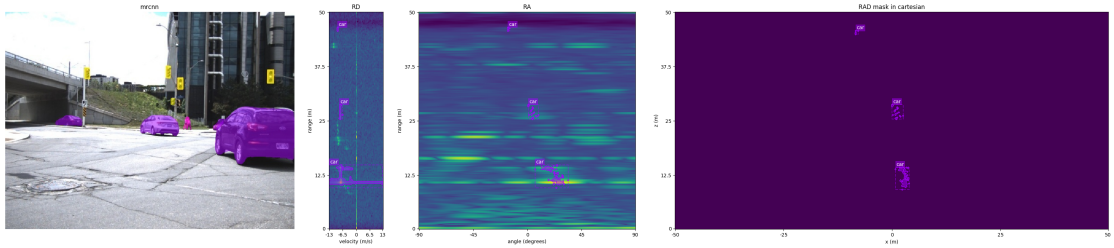

FIGURE 4.9: An example of the annotation result from our proposed method.

Various conditions have been tested with this auto-annotation algorithm during the experiments. Different road types, such as intersections and sidewalks were both tested during the data capture. As a result, our algorithm shows an overall high accuracy of annotating radar data. Meanwhile, error annotations were also found during the experiments. Several reasons are attributed to the issues:

- **FOVs**: The FOVs of both sensors are quite different. The stereo cameras have a FOV of approximately 50°, while the FOV of the radar is almost 180°. As a result, a considerable number of radar instances are left un-labelled. An example is shown as Figure 4.8.
- **Mask-RCNN False Detections**: The pre-trained Mask-RCNN used in this research has a mask AP of 39.5%. Both false category predictions and inaccurate segmentations can impact the final quality of the radar annotations.
- **Radar Ambiguity**: The detection of the radar heavily depends on the reflected signals from the objects. Either too much noise or too weak the reflected signals can be resulted when the objects are too close or too far away from the radar, and thus leads to false detections.
- **Stereo Ambiguity**: As the data capture sessions were conducted in different environments, the changing lighting and weather conditions could impact the cameras' intrinsic properties and thus lead to the noisy stereo depth estimation.

To tackle these issues, manual corrections were conducted after the auto-annotations. Therefore, high-quality annotations are guaranteed in the final radar dataset. Figure 4.12 shows some samples from our radar dataset.

4.3 Dataset Statistical Analysis

Our radar dataset contains RAD tensors as the inputs and radar annotations as the ground truth labels. The radar annotations consist of the bounding boxes on both RAD tensors and the corresponding Cartesian maps for localization, and the categories for classification. The dataset is publicly available[4] for further research in this area. Details and the corresponding statistical analysis are introduced in the following sections.

4.3.1 Overall Analysis

The dataset contains totally 6 categories, which are "person", "bicycle", "car", "motorcycle", "bus" and "truck". The 3D bounding boxes for RAD tensors consist of 3D center locations $[x_{center}, y_{center}, z_{center}]$ and sizes $[w, h, d]$, where $[w, h, d]$ means width, height and depth. Similarly, the 2D boxes contain center points and sizes in a format

[4] http://www.site.uottawa.ca/research/viva/projects/raddet/index.html

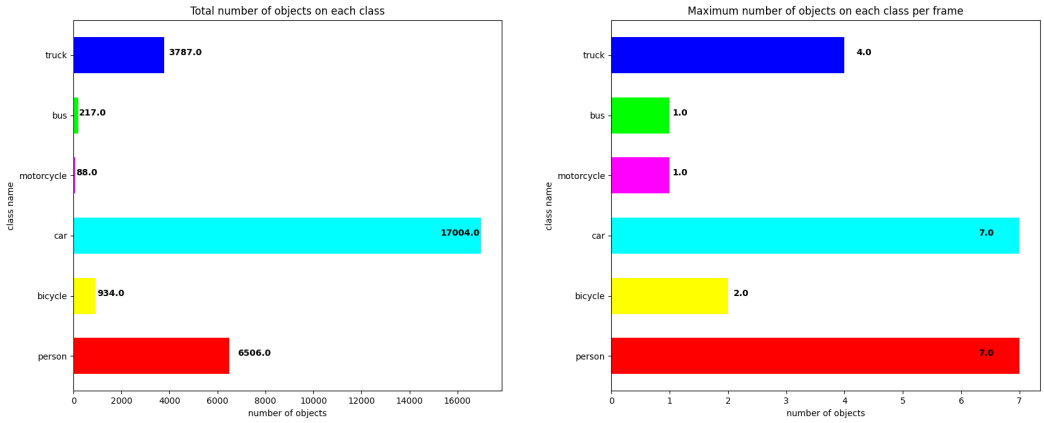

FIGURE 4.10: Left: distribution of the objects over all categories. Right: maximum number of objects that appear in a single frame.

of $[x_{center}, y_{center}, w, h]$. For visualization purpose, the stereo frames are also included in the dataset with the rectified left and right images.

Over 60800 frames were captured during one week of data capture sessions. Sensors were set up at random locations on the sidewalks facing the main roads during the data capture, as samples shown in Figure 4.12. Totally 10158 frames were selected for building the dataset. Consecutive frames were avoided during the frame selections in order to improve the dataset generalization. Manual corrections were conducted on each of the selected frames.

A systematic statistical analysis is conducted on the derived dataset. The distributions of the objects on all the categories is shown in Figure 4.10. Although class "car" dominates the entire dataset, there is still a considerable high portion of pedestrian instances included. The analysis of the object intensities within a single frame is also conducted, results are shown as the right image in Figure 4.10. The high intensities indicate that various conditions are considered in the dataset building process. The major drawback of our dataset is the data imbalance. As seen in Figure 4.10, the numbers of the objects on "bus", "motorcycle" and "bicycle" are significantly smaller than other classes. This could potentially pose a threat to the deep learning algorithms of achieving higher performance.

Since the radar is mainly a distance sensor, the range distributions of the objects on all the categories are also analyzed. Figure 4.11 shows that most objects in our dataset are distributed in a range of $10m$ to $45m$. This is due to several reasons. First, signals with high-level noises are seen in the radar frames when the objects are too close to the radar. They are filtered out during the manual corrections to avoid inaccurate annotations. Second, signals are weakened when the objects reach the maximum detection range. Thus they are hardly detectable for the radar. Despite all these issues, the range

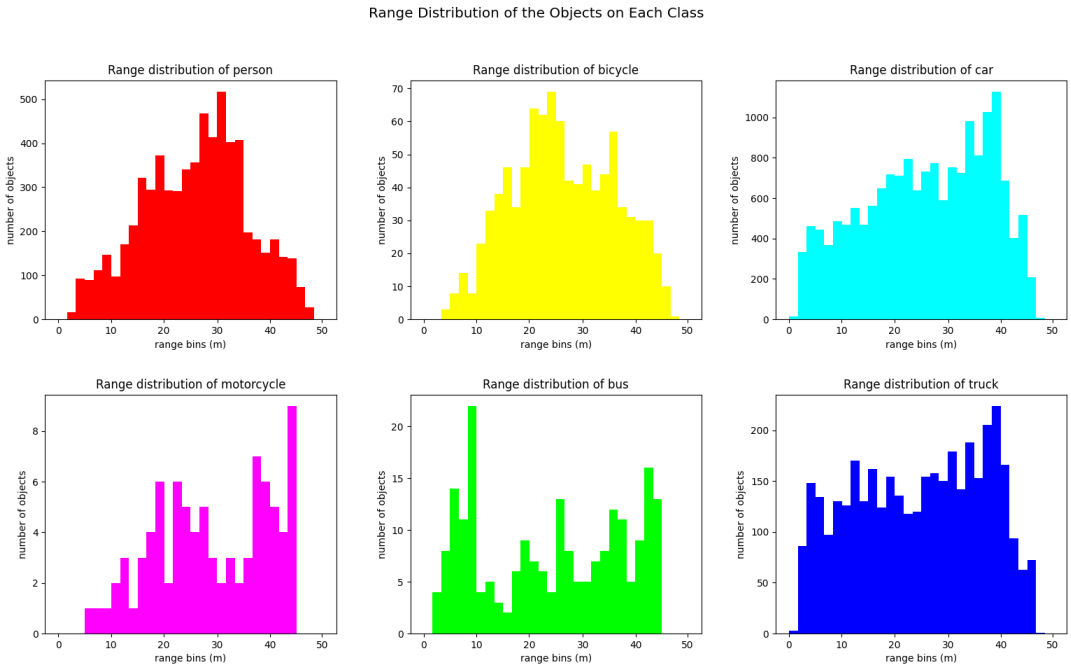

FIGURE 4.11: Range distributions of the objects on all the categories.

distributions shown in Figure 4.11 give a relatively reasonable balance over the entire detectable ranges. Thus, we consider the distributions as tolerable for radar object detection development.

4.3.2 Dataset Split

The dataset is split into train and test sets with the portions 80% and 20% for the development of our deep learning models. The splitting process strictly follows the object distributions and range distributions in the statistical analysis. Table 4.3 shows an overview of the object distributions of both train and test sets..

The train set is further split into 90% and 10% for the training and validation purposes during the model exploration and hyper-parameters tuning process. Other details are introduced in Chapter 6.

Number of Objects	**Person**	**Bicycle**	**Car**	**Motorcycle**	**Bus**	**Truck**
Train set	5210	729	13537	67	176	3042
Test set	1280	204	3377	21	38	720

TABLE 4.3: Numbers of the objects on each class of both train and test sets.

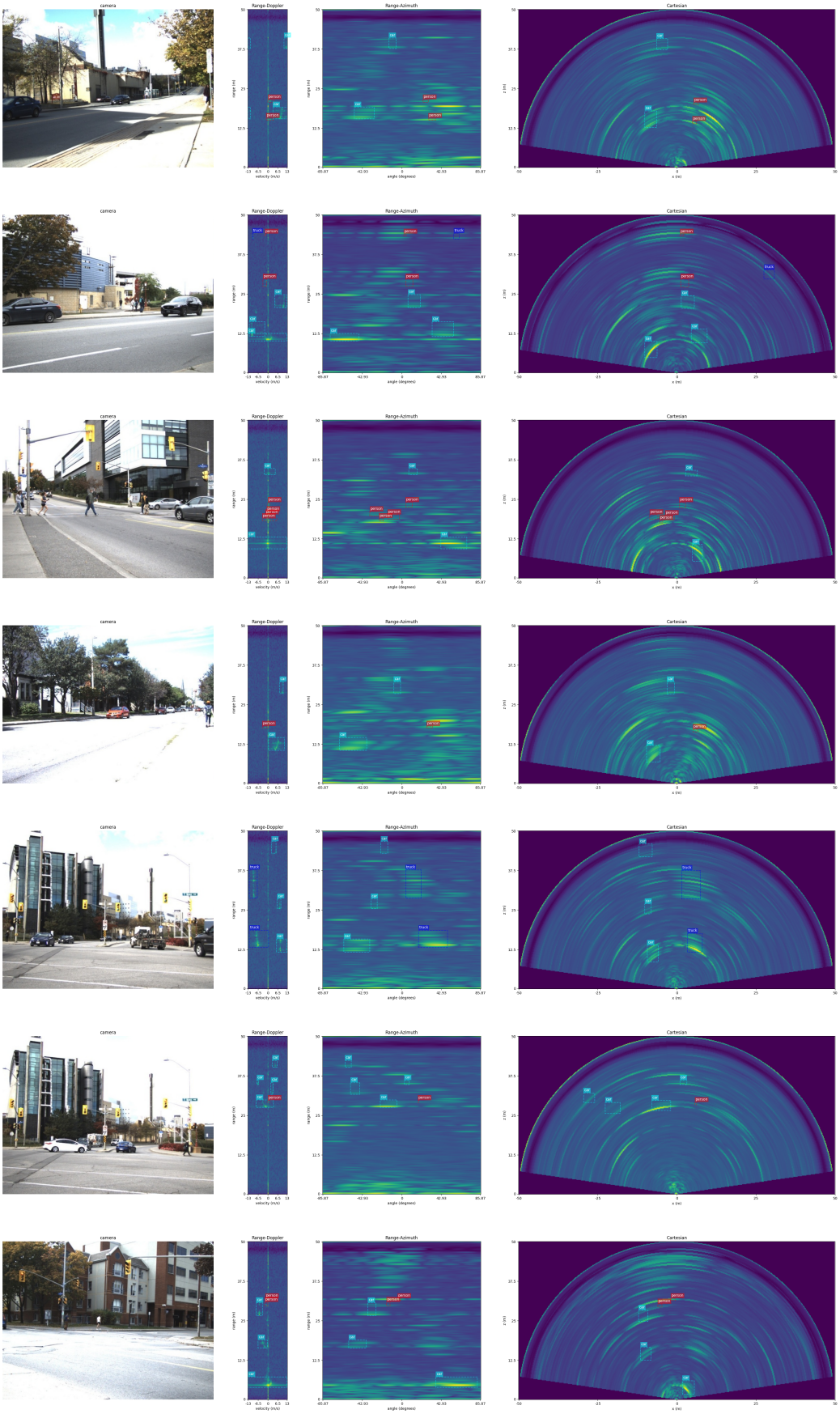

FIGURE 4.12: Samples from our proposed radar dataset.

Chapter 5

Range-Azimuth-Doppler based Object Detection

Our radar object detection model takes Range-Azimuth-Doppler (RAD) tensors as the inputs and both annotated 3D RAD instances and 2D Cartesian instances as the ground truth labels. The model contains a backbone which is developed based on a well-known architecture ResNet [25] and a dual detection head that consists of a 3D detection head and a 2D detection head for the detections on the RAD tensors and Cartesian maps respectively. Both detection heads are built on top of the YOLO [6] detection head. Figure 5.1 shows the data flow of our proposed model.

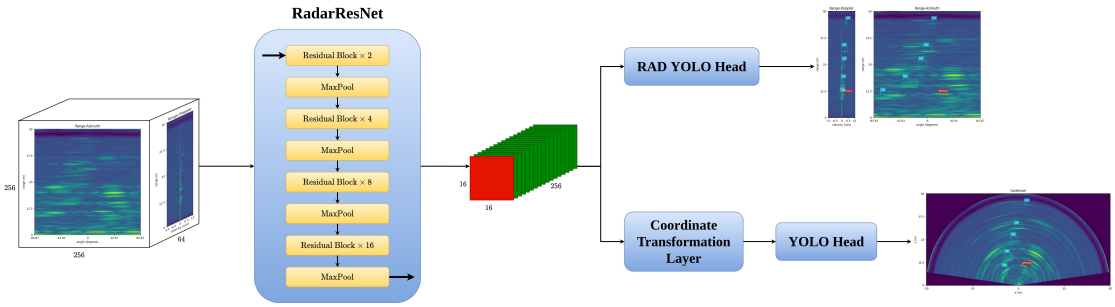

FIGURE 5.1: The overall architecture of our radar object detection model.

Unlike modern image-based object detection algorithms, our model does not have neck layers due to the following reasons. First, issues such as perspective geometry in photography do not apply to the radar domain, as radars always reveal the true sizes of the objects no matter how long the distances between the objects and the sensors. Therefore, Neck layers that are designed for resolving these issues are not necessary. Second, neck layers normally consist of extra convolutional layers, which could impact the inference speed of our model. Thus, multi-resolution neck layers are not considered in the architecture building process.

In the following sections, the input pre-processing method is first introduced before the introduction of the backbone architecture along with the corresponding operations. Then, both 3D and 2D detection heads are detailed at the end of the chapter to introduce our dual detection head.

5.1 Input

5.1.1 Input Pre-processing

Raw RAD tensors, as the products from 3D-FFT, are represented with complex numbers. Magnitude values are first extracted from the raw RAD tensors to transform those complex numbers into floats. One issue that can be found in the magnitude representations is their high-level variance. For example, the average magnitude of the detection signals is 10^2 times higher than the background signals. Therefore, log values are also extracted from those magnitude values to form the RAD inputs. The derivation of the input RAD tensors D_{RAD} from the raw RAD tensors D_{raw} is formulated as Equation 5.1.

$$I_{RADi} = \texttt{log}(\texttt{abs}(I_{rawi})), \quad I_{RADi} \in D_{RAD}, \ I_{rawi} \in D_{raw} \tag{5.1}$$

The size of the input RAD tensors is of the same size as the raw RAD tensors, which is $(256, 256, 64)$.

5.1.2 Global Normalization

A global-wise normalization is also conducted on the input RAD tensors. The mean value I_{mean} and the variance value $I_{variance}$ are searched through the entire dataset. The normalized RAD tensors D_{norm} are then computed as Equation 5.2.

$$I_{normi} = \frac{I_{RADi} - I_{mean}}{I_{variance}}, \quad I_{normi} \in D_{norm}, \ I_{RADi} \in D_{RAD} \tag{5.2}$$

Where, $I_{mean} = 3.2438$ and $I_{variance} = 10.0806$. The outputs are then sent to our backbone for the feature extractions.

5.2 Backbone

5.2.1 Architecture

The RAD inputs to our backbone are quite different from the images. Each input dimension stands for a unique property of the received signals from the radar. To adapt such inputs to 2D convolutional layers, one dimension has to be set as input channels. In our research, both RD-A and RA-D formats were tested during the experiments and the results show that RA-D format has slightly higher performance than RD-A. Therefore, we set Range-Azimuth dimensions as the input dimensions and Doppler axis as the input channels. This input arrangement also helps us to build our 2D detection head since the Cartesian maps are transformed from the RA spectrums.

For the size of the output feature maps, it directly decides the final output resolutions of the detection heads. Either too much compression or too shallow the model can impact the task performance. Experiments were conducted with multiple output sizes

and size $(16, 16, 256)$ outperformed all other tested sizes. We thus set the output size of our model as $(16, 16, 256)$.

With fixed input size and output size, the total numbers of feature-map downsamplings and channel expansions are computed as 4 and 2 respectively. Popular image-based backbones such as VGG [68] and ResNet [25] were tested along with various feature downsampling methods during the model exploration. As the numbers of the feature downsamplings and the channel expansions of the original image-based backbones are different from our research, modifications were applied to those backbones. For example, in VGG, we set the channel expansion rates as 1 for the first three blocks and removed the first pooling layer. Shown in Figure 5.1, our proposed backbone is formed with a similar architecture as ResNet. For convenience, we named our backbone RadarResNet.

The RadarResNet contains two types of blocks, namely residual block and downsampling block. The residual block has a similar structure as the basic residual block from ResNet [25]. Figure 5.2 shows the structure of our residual block. The channel expansion rates of the last residual blocks in "Residual Block $\times 8$" and "Residual Block $\times 16$" are set to 2 for expanding feature channels. The downsampling block consists of a

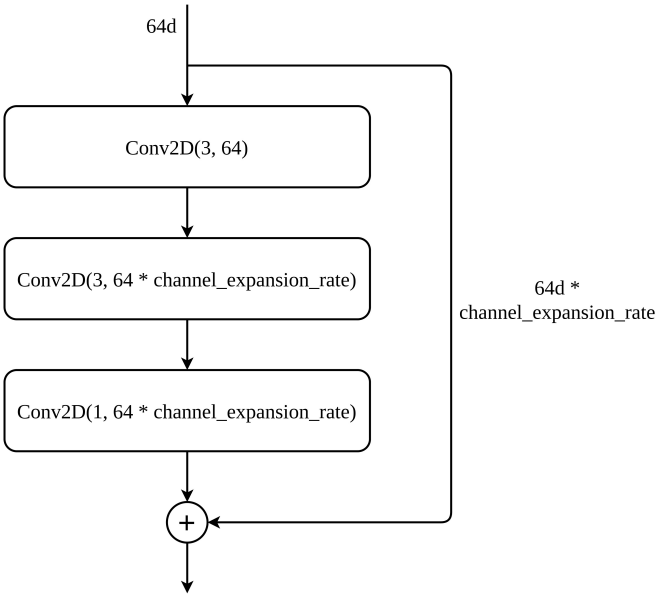

FIGURE 5.2: An example of the residual block in our model, with a similar structure as the basic residual block in ResNet [25]. The values in the brackets stand for kernel size and output channels respectively.

single max-pooling layer. It is experimentally proven that max-pooling layers can lead to higher performance than stride two convolutional layers in our research, details are introduced in Chapter 6.

Figure 5.3 shows some output feature map samples of our RadarResNet from a specific input RAD tensor. From the figure, most generated feature maps show the ability

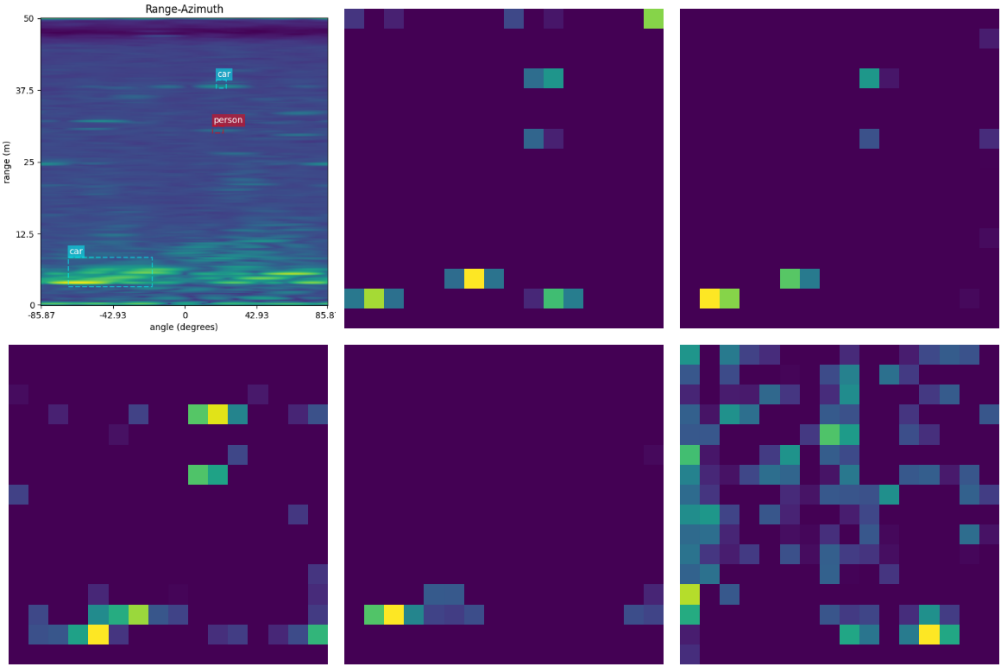

FIGURE 5.3: Output feature map samples from our backbone with top-left image illustrating the final predictions.

to identify and highlight the locations and roughly the sizes of the targeted objects on the RAD spectrum. Some feature maps, shown as the last image in Figure 5.3, even have the capacity of extracting background noise signals. This proves that our backbone, RadarResNet, is able to distinguish the objects from the background noises, and extract the features and sizes of the detected objects.

5.2.2 Operations

In the RadarResNet, bias is added to each convolutional layer in the residual blocks. Zero-padding is set to "same" for ensuring that the output size is the same as the input size for each convolutional layer. Batch Normalization (BN) [30] is added to the convolutional layers for the normalization and Rectified Linear Unit (ReLU) is set as the activation functions. In addition, L2 regularizers are added during the training to relieve the overfitting.

5.3 Dual Detection Head

Our dual detection head consists of a 3D detection head that performs object detection on the RAD tensors and a 2D detection head for the predictions on the Cartesian maps. Both detection heads are developed based on a classic one-stage anchor-based detection head, YOLO [6]. The anchor boxes of both detection heads are defined by applying K-means Clustering [1] through all the instances in our dataset. Figure 5.4 shows the

details of the anchor box finding process. The error rate threshold of the K-means Clustering algorithms was set to 10% during the experiments.

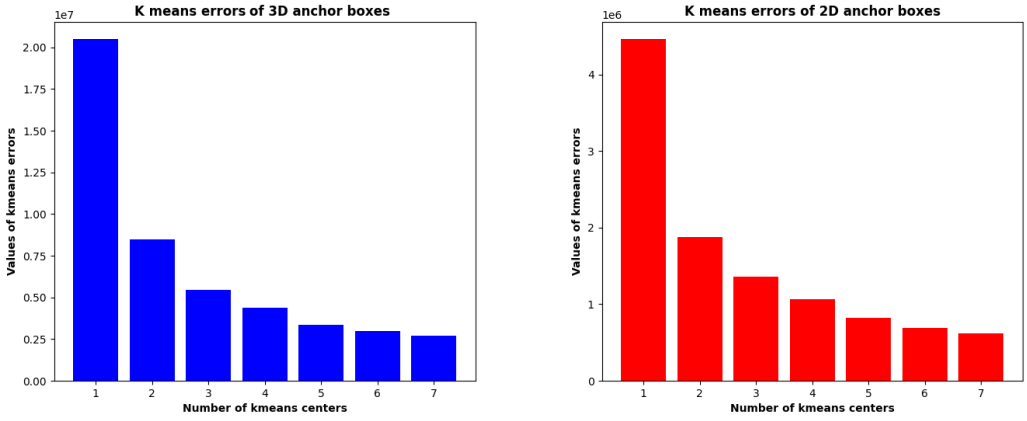

FIGURE 5.4: Left: RAD 3D anchor boxes finding process. Right: Cartesian 2D anchor boxes finding process.

As a result, six 3D anchor boxes and six 2D anchor boxes are defined as Table 5.1. Aspect ratios are not defined in our research due to the reason that the box decoding process of YOLO [6] head relies on box sizes only.

Anchor box number	**3D anchor box ($[w, h, d]$)**	**2D anchor box ($[w, h]$)**
Anchor box 1	$[6, 9, 5]$	$[11, 15]$
Anchor box 2	$[19, 36, 60]$	$[28, 24]$
Anchor box 3	$[20, 28, 19]$	$[6, 8]$
Anchor box 4	$[26, 75, 61]$	$[43, 41]$
Anchor box 5	$[22, 45, 28]$	$[17, 27]$
Anchor box 6	$[19, 17, 8]$	$[21, 18]$

TABLE 5.1: Sizes of both 3D and 2D anchor boxes. w, h, d stand for width, height, depth respectively.

5.3.1 3D Detection Head

The 3D detection head generates low-resolution 3D grid maps with the information illustrating box locations and categories. Similar to YOLO head which transforms the

input feature maps to the format $[Grid_{width}, Grid_{height}, Num_{anchors} \times (Num_{classes} + 5)]$ [6], our 3D detection head processes the raw feature maps as described below,

$$FeatureMaps \rightarrow [Grid_{width}, Grid_{height}, Grid_{depth} \times Num_{anchors} \times (Num_{classes} + 7)] \quad (5.3)$$

where, $Grid_{width}$, $Grid_{height}$ and $Grid_{depth}$ stand for the grid sizes on the Range, Azimuth and Doppler dimensions respectively. Among them, $Grid_{width}$ and $Grid_{height}$ are kept the same sizes as the raw feature maps and $Grid_{depth}$ is calculated with the same grid strides as the other two dimensions. In our case, the value of $Grid_{depth}$ is computed as 4. $Num_{anchors}$ and $Num_{classes}$ indicate the total number of anchor boxes and the number of all the categories respectively. 7 stands for an "objectness" and the raw box information, where the raw box information consists of the raw center locations $[x_r, y_r, z_r]$ and raw sizes $[w_r, h_r, d_r]$. The output of our 3D detection head is then reshaped and split according to different tasks.

The decoding process of our 3D detection head is the same as YOLO [6] head. The output is first split into objectness, raw categories and raw boxes. Then, the raw box center locations $[x_r, y_r, z_r]$ and sizes $[w_r, h_r, d_r]$ are interpreted as Equation 5.4 to Equation 5.9.

$$x = \sigma(x_r) + c_x \quad (5.4)$$

$$y = \sigma(y_r) + c_y \quad (5.5)$$

$$z = \sigma(z_r) + c_z \quad (5.6)$$

$$w = p_w \cdot e^{w_r} \quad (5.7)$$

$$h = p_h \cdot e^{h_r} \quad (5.8)$$

$$d = p_d \cdot e^{d_r} \quad (5.9)$$

Where, c_x, c_y, c_z stand for the grid map locations; $[x, y, z]$ and $[w, h, d]$ are the decoded box centers and sizes; p_w, p_h, p_d are the anchor box sizes. The raw categories are processed by SoftMax function and decoded by one-hot function. As for the objectness, Sigmoid function is used for the interpretation.

During the training, all ground truth boxes and categories are transformed to the output format described in Equation 5.3. At each grid point, the objectness is labelled as 1 if the corresponding anchor box has an Intersection of Union (IoU) over 50% with the ground truth boxes. Background grids are marked as 0. Box information and categories are then filled in to form the training labels.

As our 3D detection head is developed based on YOLO [6] head, we named it RAD YOLO Head for convenience.

5.3.2 2D Detection Head

The 2D detection head consists of a Coordinate Transformation Layer and a YOLO [6] detection head due to the reason that the Cartesian maps are generated from the RA features by using non-linear transformation.

Traditionally, the transformation of range-angle representations $[r, \theta]$ from the Polar frame to width-depth representations $[x, z]$ from the Cartesian frame is described as Equation 5.10 and Equation 5.11.

$$x = r \cdot \cos\theta \tag{5.10}$$

$$z = r \cdot \sin\theta \tag{5.11}$$

$$\theta \in [-\pi/2, \pi/2]$$

One property of this highly non-linearly transformation is that the output Cartesian maps double-size the width dimension of the input Polar maps.

Our Coordinate Transformation Layer is developed under the inspiration of the traditional transformation method. As introduced before, the raw feature maps from our backbone are sized $(16, 16, 256)$. Another interpretation of these feature maps can be seen as 256 RA maps that are sized $(16, 16)$ with the polar representations $[r, \theta]$. Therefore, the output Cartesian representations of the Coordinate Transformation Layer should be sized $(32, 16)$. For non-linear transformation, matrix multiplication is considered as the most effective way. In our research, each $(16, 16)$ RA map is flattened before being processed by the fully connected layers for the non-linear transformation. The double-sized outputs are then reshaped into the desired size $(32, 16)$ for the Cartesian interpretations. Figure 5.5 shows the data pipeline of our proposed Coordinate Transformation Layer. The multiple $(32, 16)$ output maps are then concatenated into

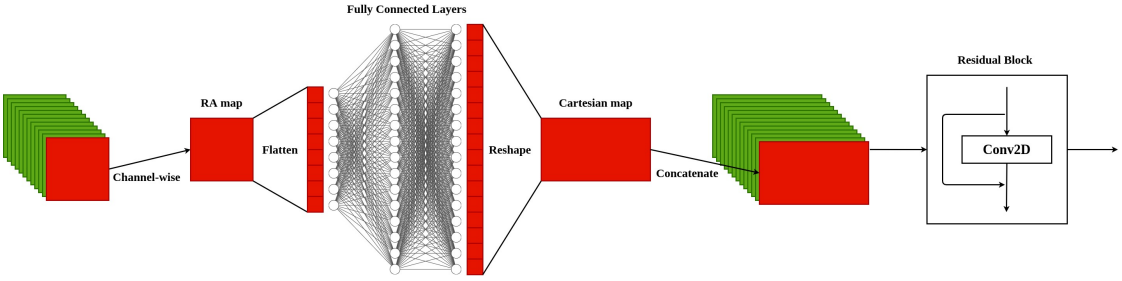

FIGURE 5.5: Coordinate Transformation Layer that consists of channel-wise MLP for non-linear transformation and a residual block for summarizing low-level features.

the feature maps with size $(32, 16, 256)$. Since these feature maps represent low-level features, a residual block, with the structure shown in Figure 5.2, is added to the end of the Coordinate Transformation Layer for the feature summarization.

Finally, a classic YOLO [6] Head is added for 2D object detection on the Cartesian maps.

5.4 Loss Functions

Based on different tasks split from YOLO [6] detection head, different loss functions are applied for the training process. Details are introduced below.

For the box regression, various loss functions have been explored over the recent years. Among them, MSE and GIoU [57] are the most popular ones. In our research, the MSE-based loss function from YOLOv2 [6] is applied for the box regressions of both RAD YOLO Head and 2D YOLO Head. The details of this function L_{box} is described as Equation 5.12.

$$\begin{aligned} L_{box} = \lambda \sum_{i=0}^{G} \sum_{j=0}^{B} \mathbb{1}_{ij}^{obj} \Big[(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 + (z_i - \hat{z}_i)^2 \Big] + \\ \lambda \sum_{i=0}^{G} \sum_{j=0}^{B} \mathbb{1}_{ij}^{obj} \Big[(\sqrt{w_i} - \sqrt{\hat{w}_i})^2 + (\sqrt{h_i} - \sqrt{\hat{h}_i})^2 + (\sqrt{d_i} - \sqrt{\hat{d}_i})^2 \Big] \end{aligned} \tag{5.12}$$

Where, x_i, y_i and z_i are the box center predictions; $\hat{x}_i$, $\hat{y}_i$ and $\hat{z}_i$ are the ground truth box center locations; w_i, h_i and d_i are the box size predictions; $\hat{w}_i$, $\hat{h}_i$ and $\hat{d}_i$ are the ground truth box sizes; G stands for the set of all grid locations; B is the set of all anchor boxes; $\mathbb{1}_{ij}^{obj}$ is the object mask with the objects as 1 and background as 0; the hyper-parameter λ is set to 1 in our research.

For the objectness, Focal Loss [37] is set as the loss function. As proven in the experiments, this type of function can accelerate the training process with much smoother convergence. The formal description of the objectness loss function L_{obj} is as Equation 5.13

$$\begin{aligned} L_{obj} = (1-\alpha) \sum_{i=0}^{G} \sum_{j=0}^{B} \mathbb{1}_{ij}^{obj} (1 - p_{obj})^{\gamma} \mathtt{CE}(1, p_{obj}) + \\ \alpha \sum_{i=0}^{G} \sum_{j=0}^{B} \mathbb{1}_{ij}^{nonobj} (0 - p_{obj})^{\gamma} \mathtt{CE}(0, p_{obj}) \end{aligned} \tag{5.13}$$

Where, $\mathbb{1}_{ij}^{obj}$ and $\mathbb{1}_{ij}^{nonobj}$ stand for object and non-object masks; p_{obj} means the prediction probability of the objectness; $\mathtt{CE}$ is Cross Entropy [46]; hyper-parameters γ and α are set to 2 and 0.01 respectively.

For the loss function of classification $L_{category}$, classic Cross Entropy [46] is used in our research, shown as Equation 5.14.

$$L_{category} = \sum_{i=0}^{G} \sum_{j=0}^{B} \mathbb{1}_{ij}^{obj} \mathtt{CE}(\hat{p}_{class}, p_{class}) \tag{5.14}$$

Where, $\hat{p}_{class}$ and p_{class} stand for ground truth and prediction category probabilities.

The total loss function L_{total} is set as a combination of the box regression loss function L_{box}, the objectness loss function L_{obj} and the category loss function $L_{category}$. During the experiments, we found that L_{box} can easily dominate the value of L_{total}. Therefore, another hyper-parameter β is added to L_{box} to relieve this issue. The description of the final total loss function is shown as Equation 5.15.

$$L_{total} = \beta L_{box} + L_{obj} + L_{category} \tag{5.15}$$

Where, β is set to 0.1 in our research.

Chapter 6

Experiments

Our experiments are designed by following the setups of image-based object detection development. Unit tests are first conducted for the input and output format searching. Then, the entire train set is used for the backbone architecture searching and hyper-parameters tuning. In order to investigate the general impacts of modern deep learning techniques such as self-attention layers on radar object detection, different models are formulated and tested on the test set for systematic comparisons. In this chapter, the training parameter setups and the unit tests are introduced first, followed by the backbone architecture exploration. Then, additional layers such as self-attention layers and channel-wise Multilayer Perceptrons (MLP) are introduced and the differences between all the tested models are reported and analyzed.

6.1 Training Setups

The training parameter setups for the experiments in our research are summarized in Table 6.1. All parameters are experimentally proven to be the most suitable for the

Parameter	Value
Batch Size	3
Warmup Learning Rate	0.000001
Initial Learning Rate	0.0001
Learning Rate Decay	0.96
Warm-up Steps	$50K$
Learning Rate Decay Steps	$10K$
Objectness threshold (3D & 2D)	0.5
NMS threshold (3D & 2D)	0.3
Optimizer	Adam
Metrics	mAP

TABLE 6.1: Training parameter settings in our research.

training process of our model. For example, the Initial Learning Rate in Table 6.1 is derived from the systematic experiments. Such a learning rate can ensure that the training

process has the fastest convergence speed with least chance of divergence. To further stabilize the training, a warmup process is also added prior to the formal training. All the experiments are conducted with an RTX 2080 Ti GPU and TensorFlow API[1].

During the experiments, all the backbones are trained and validated along with the RAD YOLO Head because the 3D detection head is more complex and has a higher capacity of extracting more detailed features. On the other hand, The training of the 2D detection head is conducted after the backbones are fine-tuned on the 3D detection head. The weights of the backbones are frozen during the 2D detection head training.

Unit tests are first conducted with some of the backbones before the formal training process. All the training parameters that are shown in Table 6.1 are defined and optimized based on the performance of the RAD YOLO Head in the unit tests. The unit tests are designed and implemented with a considerably small portion of the train set. All the unit test samples are randomly selected from the train set and overfitted on the object detection models. Apart from the training parameters, the input format and the output size are also defined and optimized based on the training behaviour of the corresponding models in the unit tests.

6.2 Input and Output Data Representations

6.2.1 Input Format Validation

As introduced before, the raw input RAD tensors to our model are represented with complex numbers. Three different data pre-processing approaches were experimented, namely magnitude extraction, log function on signal magnitudes, and global normalization on log values. For convenience, we call them RAD_{mag}, RAD_{log} and RAD_{norm}. VGG [68] was chosen as the backbone for the input format search in the unit tests.

The results of the unit tests show that: for RAD_{mag}, the training process is relatively unstable and the loss function diverges most times; for RAD_{log}, the loss function is able to converge and the result on the validation set shows promising performance after around $1K$ steps; for RAD_{norm}, the loss function converges most quickly with the highest performance on the validation set.

Therefore, the input data format of our object detection model is set to RAD_{norm}, the globally normalized log values from the magnitudes of raw RAD tensors.

6.2.2 Output Format Validation

For the output format of our dual detection head, the most decisive factor is the size of the backbone feature maps, which also influences the design of our backbone architecture. Therefore, to search the output format, we first had to find a proper feature map size. As the feature map size consists of the feature map resolution and the channel size, two experiments were conducted in order to define these two parameters.

[1] https://www.tensorflow.org/

Three resolutions were considered for the feature map resolution during the unit tests, namely $(8, 8)$, $(16, 16)$ and $(32, 32)$. VGG [68] was employed as the backbone. The test results indicate that: $(8, 8)$ leads to more false negative predictions due to its low resolution; $(32, 32)$ results in more false positive predictions as noise signals are mixed in the detections; $(16, 16)$ shows the highest performance among them. Thus, we first set the feature map resolution as $(16, 16)$.

For the channel size, four sizes, specifically 128, 256, 512, 1024, were experimented with the entire train set. Both VGG [68] and ResNet [25] were applied during the experiments. The results on the validation set show that the model starts to overfit when the channel size is larger than 256. Therefore, 256 is chosen as the final channel size.

Combining the two experiments above, the feature map size of our backbone is set as $(16, 16, 256)$.

6.3 Model Architecture Search

6.3.1 Backbone Selection

Two image-based backbones, VGG [68] and ResNet [25], were experimented in order to find a proper base model for the architecture. Since the input and output sizes are fixed as $(256, 256, 64)$ and $(16, 16, 256)$, modifications were applied to both classic backbones in order to fit our task.

In the modified VGG, channel expansion rates of the first 3 convolutional blocks are set to 1. The max-pooling layer after the first block is removed to constrain the total number of feature downsampling layers. Fully connected layers are not considered for the reason that only low-level feature maps are required for our task. Table 6.2 shows the modified version of VGG.

Blocks	**Structure**
Block 1	Conv2D(3, 64) + Conv2D(3, 64)
Block 2	Conv2D(3, 64) + Conv2D(3, 64) + MaxPool(2, 2)
Block 3	Conv2D(3, 64) + Conv2D(3, 64) + MaxPool(2, 2)
Block 4	Conv2D(3, 128) + Conv2D(3, 128) + MaxPool(2, 2)
Block 5	Conv2D(3, 256) + Conv2D(3, 256) + MaxPool(2, 2)

TABLE 6.2: Modified VGG [68] for radar object detection.

For ResNet, all the residual blocks are modified with the same structure as the basic residual block from the original ResNet [25]. Four phases are considered and the residual block repeat times are set to $[2, 4, 8, 16]$ respectively. Feature downsamplings and channel expansions are implemented into the last residual block of each phase. Figure 6.1 shows the high-level architecture of the modified ResNet.

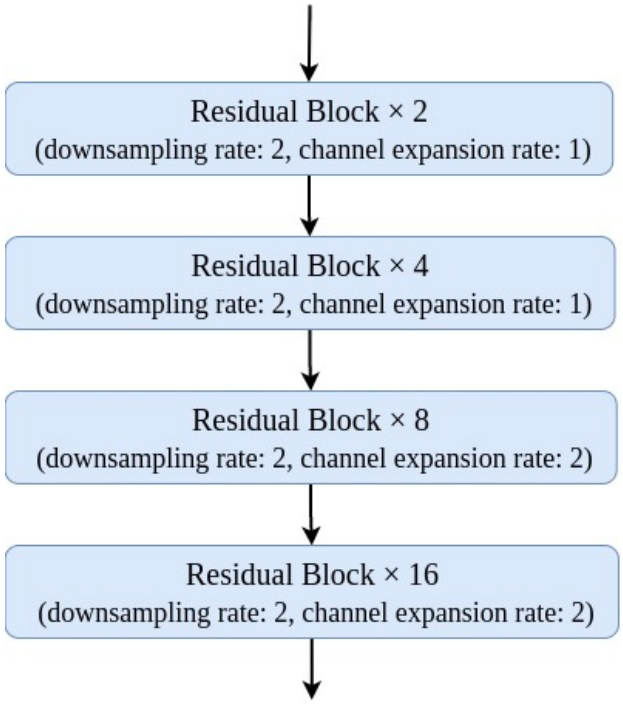

FIGURE 6.1: Modified ResNet [25] for radar object detection.

Both modified VGG and ResNet were trained and validated on the entire train set during the base model search. Since the model size of ResNet is considerably higher than VGG, different block repeat times of the 4 phases in ResNet were also explored in order to alleviate the influences of the model size. The results of the tests indicate that the performance of ResNet-based models is generally higher than VGG as the residual blocks amplify the original input signals. In order to testify the results, we also switched the convolutional blocks in VGG to the residual blocks during the experiments. The corresponding results further proved our findings. Therefore, ResNet is selected as the base model for the architecture design.

To further improve the performance of ResNet on radar object detection, different feature downsampling methods and channel expansion methods were also studied and experimented during the architecture search. Three types of feature downsampling layers, namely max-pooling layers, mean-pooling layers and stride two convolutional layers were tested individually during the tests. In addition, different locations of these downsampling layers in the backbone were also considered. For example, the downsampling layers were placed before and after each phase during the experiments in

order to study different downsampling structures. Similarly, the channel expansions were also tested with different phase locations in the modified ResNet. The results of the experiments show that, the model that uses max-pooling layer at the end of each phase for the feature downsamplings with channel expansions at the last two phases achieves the highest performance among all the tested models. We thus consider this model as our backbone architecture and named it RadarResNet, as introduced in the previous chapters.

6.3.2 Model Size

Model size is another factor that impacts not only the model performance but also the runtime speed. In our research, The size of RadarResNet is defined by the residual block repeat times of the four phases. Five sets of the block repeat times were experimented on the train set in order to optimize the model size. Details are shown in Table 6.3. As the results on the validation set suggest, the model starts to overfit when the size

Backbone Name	**AP**$_{0.1}$	**AP**$_{0.3}$	**AP**$_{0.5}$	**Backbone Params**
RadarResNet [1, 2, 4, 8]	0.766	0.554	0.238	3.68M
RadarResNet [2, 4, 4, 8]	0.752	0.555	0.247	3.91M
RadarResNet [4, 8, 8, 8]	0.753	0.557	0.247	4.70M
RadarResNet [2, 4, 8, 16]	0.764	0.563	0.251	6.73M
RadarResNet [4, 8, 8, 16]	0.766	0.563	0.247	7.21M

TABLE 6.3: Backbone size searching samples with numbers inside "[]" indicating the repeat times of residual blocks before downsampling.

becomes larger than $6.73M$. Therefore, we finalized our RadarResNet with the repeat times of each phase as 2, 4, 8, 16. Totally $6.73M$ learnable parameters are defined within our RadarResNet.

6.4 Loss Function Searching

Based on the different tasks extracted from our dual detection head, various loss functions were tested and analyzed during the experiments.

For box regression, popular image-based loss functions such as IoU, GIoU [57] and Dice [69] were first tested during the unit tests. The test results show that they are not able to stabilize the convergence of the loss function, especially at the beginning of the training process. This is due to the reason that the box location on the doppler axis is shifted when the object is moving faster than the radar's maximum detection velocity. As a result, the IoU calculation for the RAD bounding boxes becomes more unstable. In

addition, at the initial stage of the training, the IoU values of the predicted boxes to the ground truth boxes are most likely to be 0, which leads to zero gradients. This could also potentially impact the convergence of the loss function. Thus, MSE-based loss functions are preferable in our research. After a few tests with different MSE-based loss functions on the entire train set, we found that the loss function from YOLOv2 [6] leads to the highest performance. We thus set it as the loss function for the box regression.

For objectness, Cross Entropy (CE) [46] and Focal Loss [37] were tested during the experiments. As the results indicate, both functions are able to converge the loss values to the same level. Meanwhile, Focal Loss [37] takes considerably less training steps for convergence than CE. Therefore, Focal Loss [37] is defined as the loss function for the objectness.

For classification, Cross Entropy is chosen as the loss function in our research since its robustness has been proven in various applications.

All the loss function related coefficients and hyper-parameters are tuned on the entire train set. Coefficients that lead to the highest performance on the validation set are selected for the training process in our research.

6.5 Self-attention Exploration

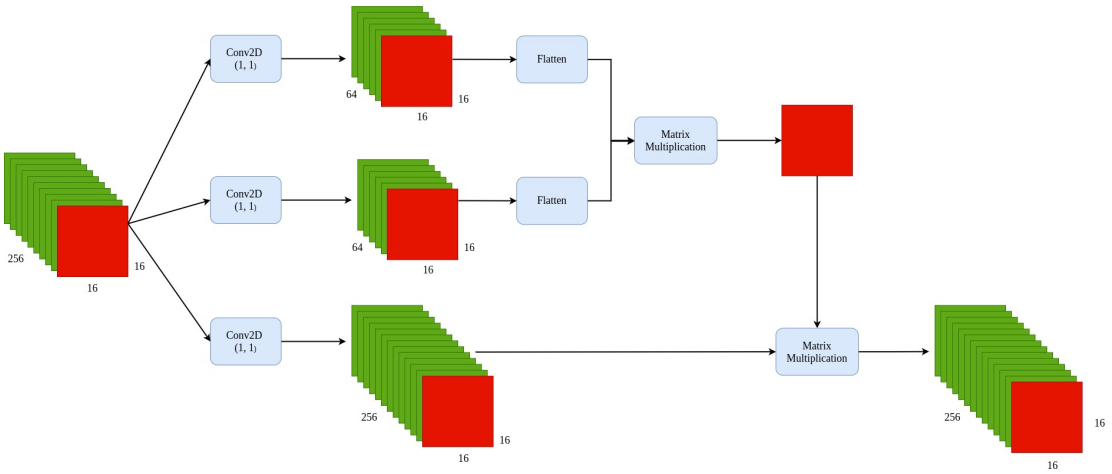

FIGURE 6.2: Modified version of the self-attention layers in SAGAN [88].

Self-attention layers, as proven in many researches [88, 49, 53, 12], have the potential to improve the performance of state-of-the-art deep learning models furthermore. The applications of self-attention layers in various tasks, such as image recognition and instance segmentation, have been studied recently. In order to study the general impacts of self-attention layers on radar object detection, two CNN-based self-attention layers from SAGAN [88] and SAUNet [49] were experimented along with our RadarResNet in this research. As they both are designed for the specific architectures, modifications were applied to both self-attention layers in order to fit our task.

6.5.1 SAGAN

For the self-attention layers in the original SAGAN [88], the attention score map is computed with matrix multiplications and the channel sizes of the hidden layers are kept the same as the size of the input feature maps. In our research, these self-attention layers were first experimented with RadarResNet in the unit tests. During the tests, we found that the training process was extremely slow with these additional layers. Therefore, some modifications were applied to the original SAGAN self-attention layers in order to improve the runtime speed. As a result, all the channel sizes of the hidden layers are switched to 25% of the original sizes. Figure 6.2 shows the implementation of SAGAN self-attention layers in our research. We named this type of self-attention layers integrated to our RadarRestNet as RadarResNet+SA (SAGAN).

6.5.2 SAUNet

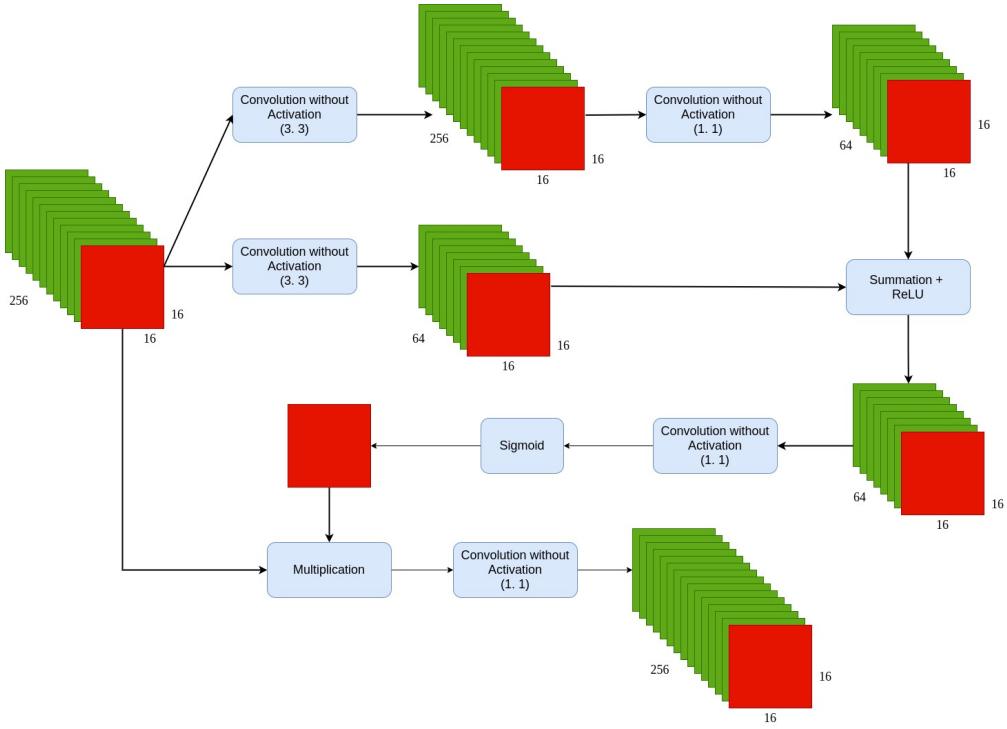

FIGURE 6.3: Modified version of the self-attention layers in SAUNet [49].

The self-attention layers in SAUNet [49], on the other hand, use pure convolutional layers for the derivation of the attention score map. Since these layers are designed specifically for the UNet [58]-like architectures, we also modified them for our research purpose. In the modified version, the input to the "gating" branch is switched to the same input as other branches and the corresponding upsampling layers are removed for the consistency of the output sizes. The channel sizes of the hidden layers are also down-sized to 25% of the original sizes. Other layers and operations are kept the same

as the original implementations in the SAUNet. Figure 6.3 shows the data pipeline of the modified version of SAUNet self-attention layers. We named this type of self-attention layers combined with our RadarRestNet as RadarResNet+SA (SAUNet).

During the experiments, RadarResNet+SA (SAGAN) and RadarResNet+SA (SAUNet) were tested individually with the self-attention layers prior to both 3D and 2D detection heads in order to investigate the impacts of them on radar object detection. Details are given in the following sections.

6.6 Systematic Comparisons

Systematic comparisons are conducted with all the explored models in our research, including modified VGG, modified ResNet, RadarResNet, RadarResNet+SA (SAGAN) and RadarResNet+SA (SAUNet). These models were trained on the entire train set and tested on the test set during the experiments. The corresponding results are compared and analyzed to provide an insight into the performance of image-based models on radar object detection. As our 3D and 2D detection heads contain the detections on two different data representations, they are introduced separately in the following sections.

6.6.1 3D Detection Head

In order to study the impact of channel-wise fully connected layers on the 3D detection head, we also applied the channel-wise MLP from our Coordinate Transformation Layer to RadarResNet to form another model. For convenience, we named it RadarResNet+MLP. Therefore, totally 6 models were tested during the experiments. The overall results and class-wise performances are shown in Table 6.4, Table 6.5 and Figure 6.4

Backbone Name	**AP**$_{0.1}$	**AP**$_{0.3}$	**AP**$_{0.5}$	**AP**$_{0.7}$	**Backbone Params**	**Inference Time**(ms)
VGG	0.685	0.499	0.209	0.035	**2.10M**	**10.5**
ResNet	0.727	0.538	0.243	0.054	6.81M	75.3
RadarResNet	**0.764**	**0.563**	**0.251**	**0.059**	6.73M	75.2
RadarResNet+SA(SAGAN [88])	0.738	0.498	0.219	0.044	4.11M	50.9
RadarResNet+SA(SAUNet [49])	0.752	0.559	**0.251**	0.053	7.55M	78.7
RadarResNet+MLP	0.759	0.521	0.217	0.043	6.86M	77.6

TABLE 6.4: Average Precisions of different backbones on RAD YOLO Head. The parameter size of the head layers is 1.34*M*.

Backbone Name	**AP**$_{person}$	**AP**$_{bicycle}$	**AP**$_{car}$	**AP**$_{motorcycle}$	**AP**$_{bus}$	**AP**$_{truck}$
VGG	0.290	0.232	0.599	0	0.368	0.441
ResNet	0.311	0.322	0.647	0.048	0.342	0.479
RadarResNet	**0.344**	0.315	**0.659**	0.095	**0.447**	0.506
RadarResNet +SA(SAGAN [88])	0.232	0.171	0.611	0.095	0.316	0.493
RadarResNet +SA(SAUNet [49])	0.350	0.295	0.653	0.048	**0.447**	**0.512**
RadarResNet+MLP	0.286	**0.353**	0.614	0.095	0.443	0.507

TABLE 6.5: Class-wise Average Precisions of different backbones on RAD YOLO Head with IoU threshold 0.3.

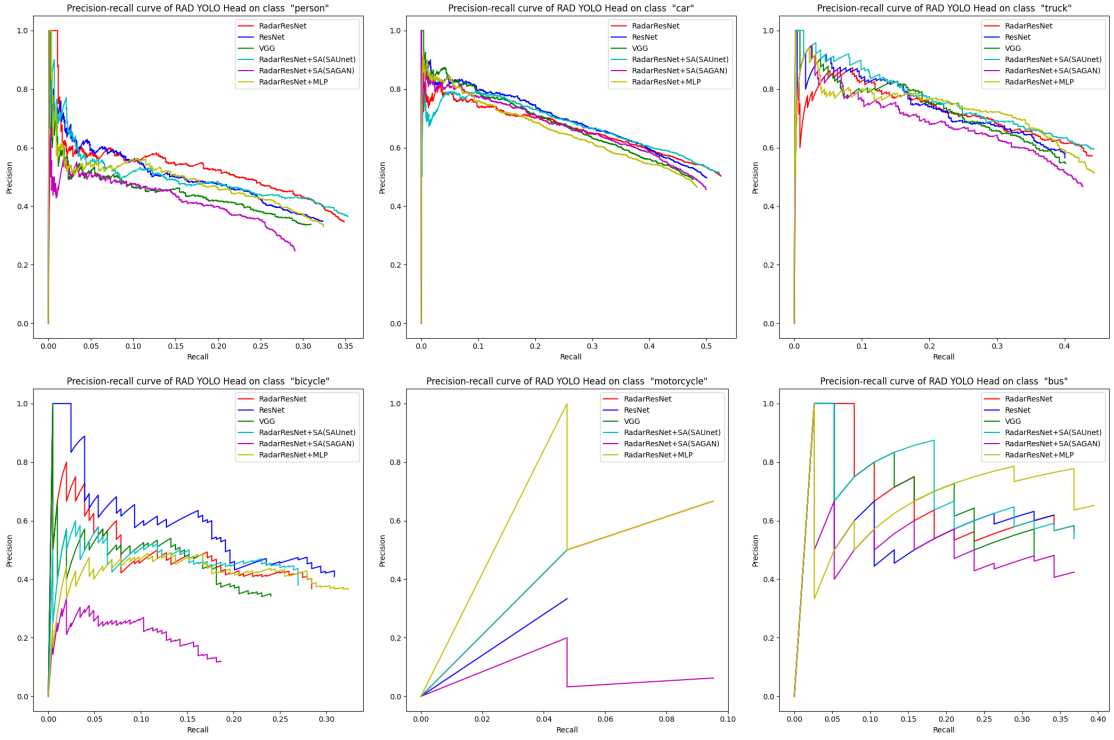

FIGURE 6.4: Class-wise precision-recall curves of different backbones on RAD YOLO Head with IoU threshold 0.3.

Among them, VGG shows the highest inference speed with the smallest model size. The higher AP of ResNet than VGG proves that residual blocks can boost the performance on both image object detection and radar object detection, since the input signals are amplified during the process. For RadarResNet and ResNet, the results show that max-pooling layers are more effective than stride two convolutional layers for the feature downsampling. Although RadarResNet+SA (SAGAN) and RadarResNet+SA

(SAUNet) show their potential of achieving high precision, RadarResNet still dominates the performance. This is due to the reason that the self-attention layers in both RadarResNet+SA (SAGAN) and RadarResNet+SA (SAUNet) are specifically designed for image-based tasks and may not be suitable for radar object detection. As for RadarResNet+MLP, the results show that the performance of the channel-wise MLP is very limited and thus it may not be feasible for the RAD YOLO Head. The low precisions of all the models on the categories "motorcycle" and "bicycle" indicate that the imbalanced dataset can impact the model performance.

The overall high performance shows that image-based object detection and radar-based object detection share many similarities. Thus, more image-based techniques can be explored in future researches for higher performance and faster speeds.

6.6.2 2D Detection Head

After the experiments on the 3D detection head, we froze the backbone of each model before training the 2D detection head. As the 2D detection head contains a Coordinate Transformation Layer, its size is larger than the RAD YOLO Head. And thus, the inference speed of the 2D detection head is slightly slower than the 3D detection head. In addition, since the 2D detection head already contains fully connected layers, RadarResNet+MLP was not considered in the 2D detection experiments. Table 6.6, Table 6.7 and Figure 6.5 show the results of the experiments.

Backbone Name	**AP**$_{0.1}$	**AP**$_{0.3}$	**AP**$_{0.5}$	**AP**$_{0.7}$	**Backbone Params**	**Inference Time**(ms)
VGG	0.757	0.685	0.484	0.189	**2.10M**	**12.8**
ResNet	0.770	0.699	0.502	0.198	6.81M	76.7
RadarResNet	0.796	0.727	0.516	**0.205**	6.73M	76.8
RadarResNet+SA(SAGAN [88])	0.787	0.686	0.452	0.145	4.11M	52.6
RadarResNet+SA(SAUNet [49])	**0.801**	**0.730**	**0.530**	0.202	7.55M	79.1

TABLE 6.6: Average Precisions of different backbones on 2D YOLO Head. The parameter size of the head layers is 2.86M.

The overall performance of the models on the 2D detection head shows similar trends with their performance on the 3D detection head. One difference is that the performance gap between RadarResNet+SA(SAUNet) and RadarResNet becomes much smaller, and RadarResNet+SA(SAUNet) even outperforms RadarResNet on $AP_{0.1}$, $AP_{0.3}$ and $AP_{0.5}$. This further indicates that self-attention layers have the potential for improving the general performance of deep learning models on radar object detection. Therefore, task-targeted exploration of self-attention layers can be conducted in future researches for further improvements.

Backbone Name	**AP**$_{person}$	**AP**$_{bicycle}$	**AP**$_{car}$	**AP**$_{motorcycle}$	**AP**$_{bus}$	**AP**$_{truck}$
VGG	0.197	0.204	0.597	0.048	0.395	0.494
ResNet	0.204	0.212	0.622	0	**0.553**	0.494
RadarResNet	**0.245**	0.225	0.622	0	0.447	**0.560**
RadarResNet +SA(SAGAN [88])	0.146	0.199	0.568	0.048	**0.553**	0.491
RadarResNet +SA(SAUNet [49])	0.232	**0.254**	**0.651**	0	0.474	0.538

TABLE 6.7: Class-wise Average Precisions of different backbones on 2D YOLO Head with IoU threshold 0.5.

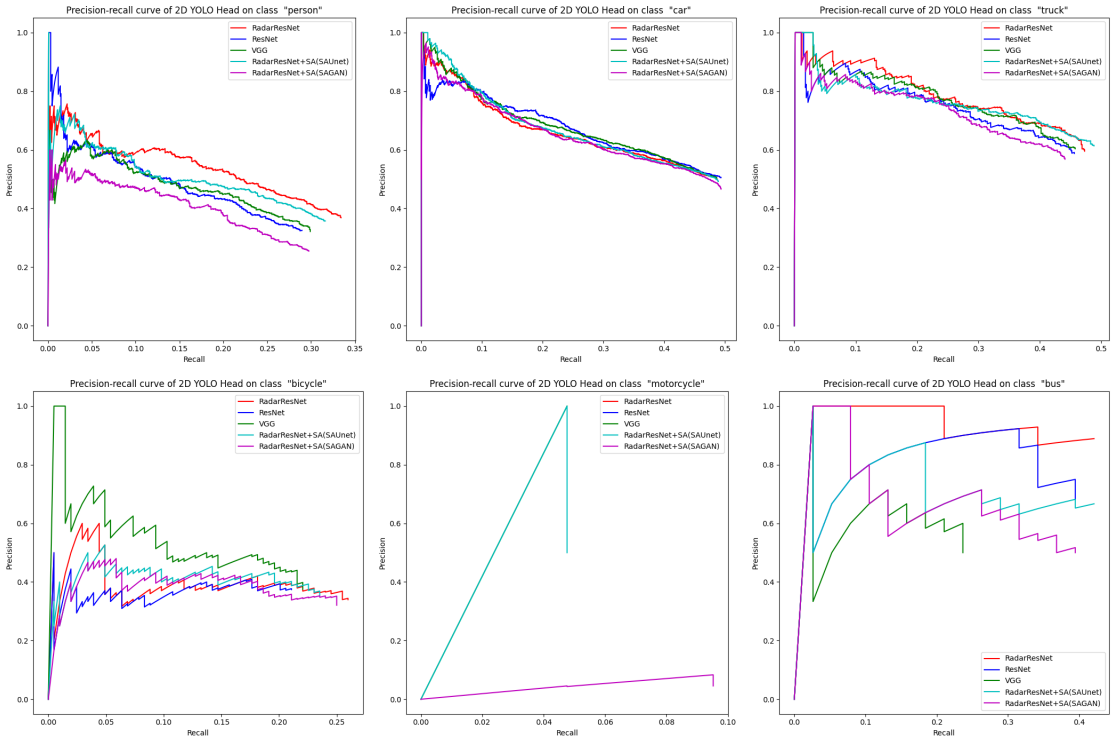

FIGURE 6.5: Class-wise precision-recall curves of different backbones on 2D YOLO Head with IoU threshold 0.5.

6.7 Discussion

Visual inspections were also conducted during the experiments. RadarResNet was applied on the entire test set with the predictions and the ground truth labels plotted on the input data. Figure 6.6 shows some samples of the model performance. Several issues were found and analyzed during the inspections. They are summarized as below.

- **Data Imbalance:** The performance of RadarResNet on different categories varies. For example, it has a very good detection rate of the category "car", while false

detections are commonly seen on the class "bicycle" and the class "motorcycle". This is caused by the data imbalance of the dataset. As introduced before, the objects are unevenly distributed over all the categories. 17004 cars are included in the dataset while the number of motorcycles is only 88.

- **Category Confusion:** For some classes, RadarResNet can hardly distinguish them and thus leads to mislabelling. This is caused by the ambiguity of the class assignments. For instance, assigning "truck" to small minivans and "car" to large SUVs can confuse the model due to the fact that there is no significant difference between the signal tracks of these objects in radar frames.

- **Noise Filtering:** The model tends to generate false positive detections on the noise signals. Two potential reasons can lead to this issue. First, background objects such as traffic signs may exhibit similar signal tracks as some targeted objects such as pedestrians in radar frames. Second, it also indicates that the model has a limited ability of noise filtering and thus model improvement in this area should be considered in the future.

Despite all the issues mentioned above, RadarResNet still has a considerably reasonable performance on the test set. The durability of our model on radar object detection shows that similarities can be found between image object detection and radar object detection. Thus, more advanced image-based algorithms can be explored in the future for the further improvement of the models on radar object detection.

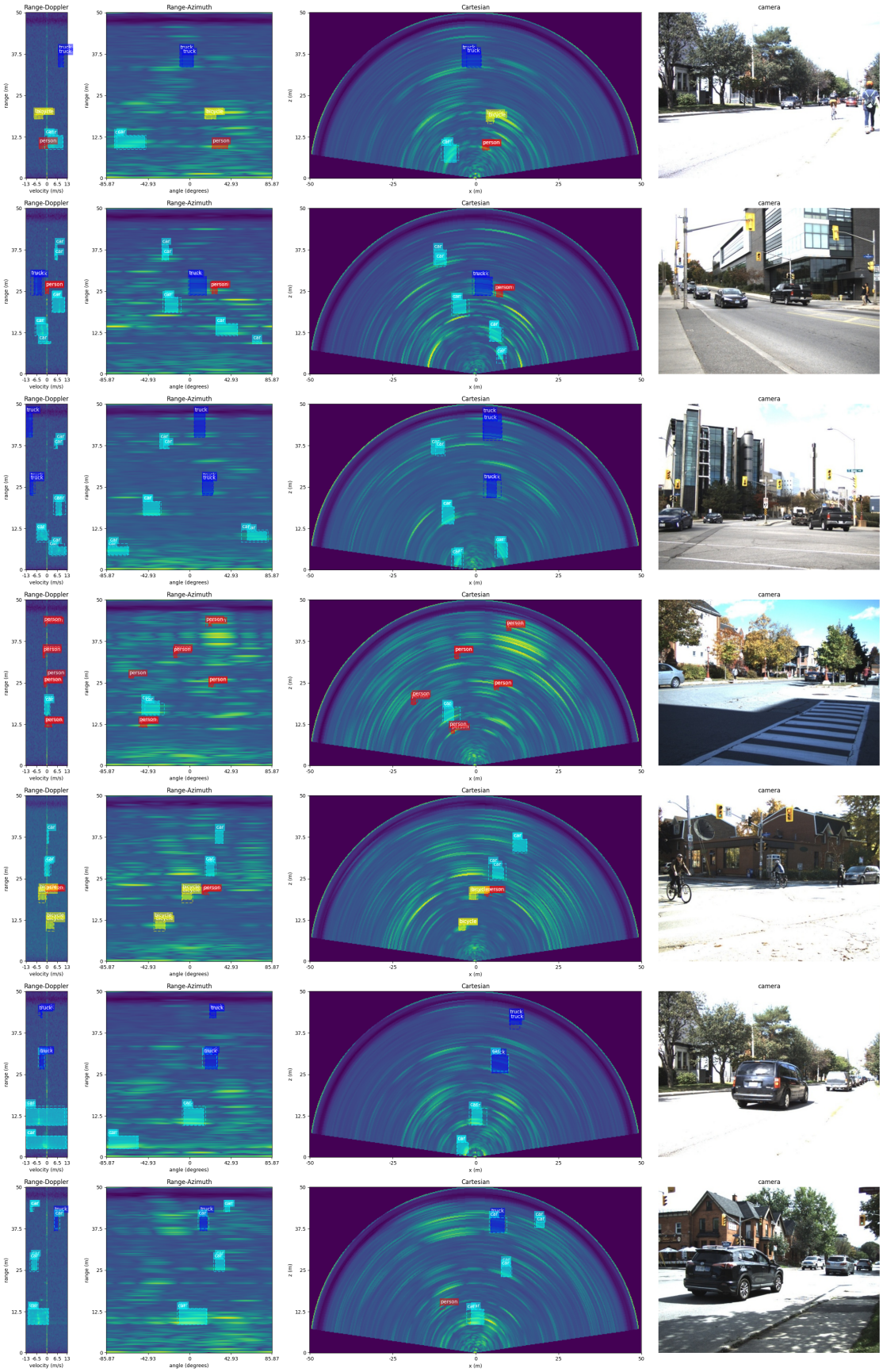

FIGURE 6.6: Samples of our model on the test set. Ground truth labels are plotted with facecolors and predictions are without facecolors.

Chapter 7

Conclusion and Future Work

7.1 Conclusion

In this research, we developed an auto-annotation algorithm for annotating radar Range-Azimuth-Doppler (RAD) tensors and their corresponding Cartesian representations. The algorithm takes full advantage of the radar's range resolution and views all dynamic road users as rigid bodies. With the help of the morphological extension, the detections from the RAD tensor are enriched and instances are extracted from those detections. Stereo vision is applied for generating class information for those radar instances. Instance-wise point clouds are generated with an instance segmentation algorithm, Mask-RCNN [24], during the process. These point clouds are then transferred to the radar frame with the projection matrix derived from the sensor registration process. Finally, all radar instances are annotated with those projected stereo instances.

Our radar dataset has been built with this auto-annotation algorithm and manual corrections. A total of 10158 frames have been annotated and manually corrected during the dataset building process. Six categories of dynamic road users were considered, namely "person", "bicycle", "car", "motorcycle", "bus" and "truck". Statistical analysis has also been conducted on the dataset and the results show relatively acceptable range distributions of the objects over each class.

The RAD object detection model in this research has been developed with our radar dataset. It consists of a backbone RadarResNet and a dual detection head. The backbone has been designed based on a classic backbone ResNet [25] with systematic model explorations on various image-based models. The dual detection head contains a 3D detection head and a 2D detection head. Both detection heads have been developed on top of YOLO [54] detection head. In addition, the 2D detection head contains a Coordinate Transformation Layer that uses channel-wise fully connected layers to transform low-level RA features to Cartesian features. Our model achieves 56.3% AP with IoU of 0.3 on the 3D detection and 51.6% AP with IoU of 0.5 on the 2D detection. To our best knowledge, our model is the first model that takes RAD tensors as the inputs and predicts the positions and velocities of dynamic road users on both RAD tensors and Cartesian representations. Self-attention layer, as another popular image-based deep learning algorithm, has also been experimented with our model. Results show that this

type of algorithm has the potential to further improve the performance of our model on radar object detection. The development of our model also proves the feasibility of image-based deep learning algorithms adapted to radar object detection. Therefore, we believe better performance could be obtained in future researches with larger model explorations.

7.2 Future Work

Despite the high performance of our model, several issues were also identified during the experiments. Therefore, improvements need to be applied to future radar object detection development.

One of the biggest issues in our research is data imbalance. This often leads to false detections on some specific classes such as “bus” and “motorcycle”. Therefore, a class-targeted dataset development should be conducted in the future to improve the model performance on those targeted categories. In addition, although the dataset contains a large number of objects, it still has a limited size when compared to its counterparts in the image domain. Thus, more data capture and data annotation sessions need to be done in the future to enlarge the dataset.

As the objective of our research is to explore the potential of radar object detection models for the applications such as autonomous driving systems, an ego-motion based dataset development should also be conducted in the future in order to improve the practicality of our model. Therefore, ego-motion based raw radar observations, such as radar ADC and RAD data, need to be captured and annotated to build the corresponding dataset. We also believe that such a dataset can make a contribution to the safety development of self-driving vehicles.

Fast runtime speeds are normally required for the radar related developments. Although our model shows high inference speeds on an RTX 2080ti GPU, it will be slowed down when transferred to edge devices. Therefore, hardware-aware model exploration is required in future researches for the runtime speed improvement.

For further performance improvement, image-based models and algorithms need to be adjusted to target radar object detection. For example, new approaches need to be explored for computing the attention score maps in self-attention layers, in order to develop a type of radar-targeted self-attention layers that can boost the accuracy of the model. On the other hand, In the image domain, the general performance of video-based object detection algorithms is higher than single-frame based object detection methods. Therefore, for the radar object detection, sequential-frames based algorithms should also be explored for the potential performance improvement.

Bibliography

[1] D. Arthur and S. Vassilvitskii. "K-Means++: The Advantages of Careful Seeding". In: *Society for Industrial and Applied Mathematics* (2007), pp. 1027–1035.

[2] A. Bochkovskiy, C. Y. Wang, and H. M. Liao. "Yolov4: Optimal speed and accuracy of object detection". In: *arXiv preprint arXiv:2004.10934* (2020).

[3] Daniel Bolya et al. "Yolact: Real-time instance segmentation". In: *IEEE/CVF International Conference on Computer Vision*. 2019, pp. 9157–9166.

[4] Leo Breiman. "Random Forests". English. In: *Machine Learning* 45.1 (2001), pp. 5–32.

[5] D. Brodeski, I. Bilik, and R. Giryes. "Deep Radar Detector". In: *IEEE Radar Conference*. 2019.

[6] Yang-Lang Chang et al. "Ship detection based on YOLOv2 for SAR imagery". In: *Remote Sensing* 11.7 (2019), p. 786.

[7] Liang-Chieh Chen et al. "Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs". In: *IEEE transactions on pattern analysis and machine intelligence* 40.4 (2017), pp. 834–848.

[8] Liang-Chieh Chen et al. "Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs". In: *IEEE transactions on pattern analysis and machine intelligence* 40.4 (2017), pp. 834–848.

[9] *Continuous Wave Radar*. https://fas.org/man/dod-101/navy/docs/es310/cwradar/cwradar.htm. Accessed: 2021-02-11.

[10] J. Domhof, J. F. P. Kooij, and D. M. Gavrila. "An Extrinsic Calibration Tool for Radar, Camera and Lidar". In: *2019 International Conference on Robotics and Automation (ICRA)*. 2019, pp. 8107–8113.

[11] X. Dong et al. "Probabilistic Oriented Object Detection in Automotive Radar". In: *IEEE Conference on Computer Vision and Pattern Recognition Workshops*. 2020, pp. 458–467.

[12] Alexey Dosovitskiy et al. "An image is worth 16x16 words: Transformers for image recognition at scale". In: *arXiv preprint arXiv:2010.11929* (2020).

[13] Kaiwen Duan et al. "Centernet: Keypoint triplets for object detection". In: *IEEE/CVF International Conference on Computer Vision*. 2019, pp. 6569–6578.

[14] R. B. Dybdal. "Radar cross section measurements". In: *IEEE* 75.4 (1987), pp. 498–516.

[15] M. Ester et al. "A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise". In: *International Conference on Knowledge Discovery*. 1996.

[16] J. M. Garcia et al. "Identification of Ghost Moving Detections in Automotive Scenarios with Deep Learning". In: *IEEE MTT-S International Conference on Microwaves for Intelligent Mobility*. 2019, pp. 1–4.

[17] Andreas Geiger, Philip Lenz, and Raquel Urtasun. "Are we ready for Autonomous Driving? The KITTI Vision Benchmark Suite". In: *Conference on Computer Vision and Pattern Recognition*. 2012.

[18] Golnaz Ghiasi, Tsung-Yi Lin, and Quoc V Le. "Nas-fpn: Learning scalable feature pyramid architecture for object detection". In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2019, pp. 7036–7045.

[19] Ross Girshick. "Fast r-cnn". In: *IEEE international conference on computer vision*. 2015, pp. 1440–1448.

[20] Ross Girshick et al. "Rich feature hierarchies for accurate object detection and semantic segmentation". In: *IEEE conference on computer vision and pattern recognition*. 2014, pp. 580–587.

[21] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. http://www.deeplearningbook.org. MIT Press, 2016.

[22] J. Guan et al. "Through Fog High-Resolution Imaging Using Millimeter Wave Radar". In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, pp. 11461–11470.

[23] Abdul Mueed Hafiz and Ghulam Mohiuddin Bhat. "A Survey on Instance Segmentation: State of the art". In: *arXiv preprint arXiv:2007.00047* (2020).

[24] K. He et al. "Mask r-cnn". In: *IEEE International Conference on Computer Vision*. 2017, pp. 2961–2969.

[25] Kaiming He et al. "Deep residual learning for image recognition". In: *IEEE conference on computer vision and pattern recognition*. 2016, pp. 770–778.

[26] Kaiming He et al. "Spatial pyramid pooling in deep convolutional networks for visual recognition". In: *IEEE transactions on pattern analysis and machine intelligence* 37.9 (2015), pp. 1904–1916.

[27] H. Hirschmuller. "Stereo Processing by Semiglobal Matching and Mutual Information". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 30.2 (2008), pp. 328–341.

[28] Sepp Hochreiter and Jürgen Schmidhuber. "Long Short-Term Memory". In: *Neural Computation* 9.8 (1997), pp. 1735–1780.

[29] Andrew G Howard et al. "Mobilenets: Efficient convolutional neural networks for mobile vision applications". In: *arXiv preprint arXiv:1704.04861* (2017).

[30] S. Ioffe and C. Szegedy. "Batch normalization: Accelerating deep network training by reducing internal covariate shift". In: *International Conference on Machine Learning*. PMLR. 2015, pp. 448–456.

[31] Takeo Kanade, Chuck Thorpe, and William Whittaker. "Autonomous Land Vehicle Project at CMU". In: *ACM Fourteenth Annual Conference on Computer Science*. Cincinnati, Ohio, USA: ACM, 1986, pp. 71–80.

[32] Jack Kiefer, Jacob Wolfowitz, et al. "Stochastic estimation of the maximum of a regression function". In: *The Annals of Mathematical Statistics* 23.3 (1952), pp. 462–466.

[33] I. V. Komarov and S. M. Smolskiy. *Fundamentals of Short-Range FM Radar*. McGraw-Hill Professional, 2005.

[34] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. "Imagenet classification with deep convolutional neural networks". In: *Advances in neural information processing systems* 25 (2012), pp. 1097–1105.

[35] Hei Law and Jia Deng. "Cornernet: Detecting objects as paired keypoints". In: *European conference on computer vision (ECCV)*. 2018, pp. 734–750.

[36] T. Y. Lim et al. "Radar and camera early fusion for vehicle detection in advanced driver assistance systems". In: *Neural Information Processing Systems*. 2019.

[37] T. Y. Lin et al. "Focal Loss for Dense Object Detection". In: *IEEE International Conference on Computer Vision*. 2017, pp. 2999–3007.

[38] Tsung-Yi Lin et al. "Feature pyramid networks for object detection". In: *IEEE conference on computer vision and pattern recognition*. 2017, pp. 2117–2125.

[39] Tsung-Yi Lin et al. "Microsoft COCO: Common Objects in Context". In: *arXiv preprint arXiv:1405.0312* (2014).

[40] Shu Liu et al. "Path aggregation network for instance segmentation". In: *IEEE conference on computer vision and pattern recognition*. 2018, pp. 8759–8768.

[41] W. Liu et al. "Ssd: Single shot multibox detector". In: *European Conference on Computer Vision*. Springer. 2016, pp. 21–37.

[42] Alex Lubben. "Self-driving Uber killed a pedestrian as human safety driver watched". In: *Vice News* (2018). URL: https://www.vice.com/en_us/article/kzxq3y/self-driving-uber-killed-a-pedestrian-as-human-safety-driver-watched.

[43] B. Major et al. "Vehicle Detection With Automotive Radar Using Deep Learning on Range-Azimuth-Doppler Tensors". In: *IEEE International Conference on Computer Vision Workshop*. 2019, pp. 924–932.

[44] Diganta Misra. "Mish: A self regularized non-monotonic activation function". In: *arXiv preprint arXiv:1908.08681* (2019).

[45] M. Mostajabi et al. "High Resolution Radar Dataset for Semi-Supervised Learning of Dynamic Objects". In: *IEEE Conference on Computer Vision and Pattern Recognition Workshops*. 2020.

[46] G. E. Nasr, E. A. Badr, and C. Joun. "Cross Entropy Error Function in Neural Networks: Forecasting Gasoline Demand". In: *International Florida Artificial Intelligence Research Society Conference*. Association for the Advancement of Artificial Intelligence, 2002, pp. 381–384.

[47] Ghina el Natour et al. "Toward 3D Reconstruction of Outdoor Scenes Using an MMW Radar and a Monocular Vision Sensor". In: *Sensors* 15 (Oct. 2015), pp. 25937–25967.

[48] F. E. Nowruzi et al. "Deep Open Space Segmentation using Automotive Radar". In: *IEEE MTT-S International Conference on Microwaves for Intelligent Mobility*. 2020, pp. 1–4.

[49] O. Oktay et al. "Attention u-net: Learning where to look for the pancreas". In: *arXiv preprint arXiv:1804.03999* (2018).

[50] A. Palffy et al. "CNN Based Road User Detection Using the 3D Radar Cube". In: *IEEE Robotics and Automation Letters* 5.2 (2020), pp. 1263–1270.

[51] K. Patel et al. "Deep Learning-based Object Classification on Automotive Radar Spectra". In: *IEEE Radar Conference*. 2019.

[52] C. R. Qi et al. "PointNet++ deep hierarchical feature learning on point sets in a metric space". In: *Neural Information Processing Systems*. 2017, pp. 5105–5114.

[53] Prajit Ramachandran et al. "Stand-alone self-attention in vision models". In: *arXiv preprint arXiv:1906.05909* (2019).

[54] J. Redmon et al. "You only look once: Unified, real-time object detection". In: *IEEE conference on Computer Vision and Pattern Recognition*. 2016, pp. 779–788.

[55] Joseph Redmon and Ali Farhadi. "Yolov3: An incremental improvement". In: *arXiv preprint arXiv:1804.02767* (2018).

[56] Shaoqing Ren et al. "Faster r-cnn: Towards real-time object detection with region proposal networks". In: *arXiv preprint arXiv:1506.01497* (2015).

[57] Hamid Rezatofighi et al. "Generalized intersection over union: A metric and a loss for bounding box regression". In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2019, pp. 658–666.

[58] O. Ronneberger, P. Fischer, and T. Brox. "U-Net: Convolutional Networks for Biomedical Image Segmentation". In: *Medical Image Computing and Computer-Assisted Intervention*. Vol. 9351. Oct. 2015, pp. 234–241.

[59] L. A. Rosero and F. S. Osório. "Calibration and multi-sensor fusion for on-road obstacle detection". In: *Latin American Robotics Symposium and 2017 Brazilian Symposium on Robotics*. 2017, pp. 1–6.

[60] Olga Russakovsky et al. "ImageNet Large Scale Visual Recognition Challenge". In: *International Journal of Computer Vision* 115.3 (2015), pp. 211–252.

[61] Nicolas Scheiner et al. "Radar-based Feature Design and Multiclass Classification for Road User Recognition". In: *IEEE Intelligent Vehicles Symposium* (2018).

[62] Nicolas Scheiner et al. "Radar-based Road User Classification and Novelty Detection with Recurrent Neural Network Ensembles". In: *IEEE Intelligent Vehicles Symposium* (2019).

[63] Jürgen Schmidhuber. "Deep learning in neural networks: An overview". In: *Neural networks* 61 (2015), pp. 85–117.

[64] R. Schmidt. "Multiple emitter location and signal parameter estimation". In: *IEEE Transactions on Antennas and Propagation* 34.3 (1986), pp. 276–280.

[65] O. Schumann et al. "Comparison of random forest and long short-term memory network performances in classification tasks using radar". In: *Sensor Data Fusion: Trends, Solutions, Applications*. 2017, pp. 1–6.

[66] O. Schumann et al. "Semantic Segmentation on Radar Point Clouds". In: *International Conference on Information Fusion*. 2018, pp. 2179–2186.

[67] E. Shelhamer, J. Long, and T. Darrell. "Fully Convolutional Networks for Semantic Segmentation". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 39 (May 2016), pp. 1–1.

[68] K. Simonyan and A. Zisserman. "Very Deep Convolutional Networks for Large-Scale Image Recognition". In: *IEEE conference on Computer Vision and Pattern Recognition* (2014).

[69] Carole H Sudre et al. "Generalised dice overlap as a deep learning loss function for highly unbalanced segmentations". In: *Deep learning in medical image analysis and multimodal learning for clinical decision support*. Springer, 2017, pp. 240–248.

[70] Suleyman Suleymanov. "DESIGN AND IMPLEMENTATION OF AN FMCW RADAR SIGNAL PROCESSING MODULE FOR AUTOMOTIVE APPLICATIONS". MA thesis. University of Twente, Aug. 2016.

[71] Christian Szegedy et al. "Going deeper with convolutions". In: *IEEE conference on computer vision and pattern recognition*. 2015, pp. 1–9.

[72] Mingxing Tan and Quoc Le. "Efficientnet: Rethinking model scaling for convolutional neural networks". In: *International Conference on Machine Learning*. PMLR. 2019, pp. 6105–6114.

[73] Mingxing Tan, Ruoming Pang, and Quoc V Le. "Efficientdet: Scalable and efficient object detection". In: *IEEE/CVF conference on computer vision and pattern recognition*. 2020, pp. 10781–10790.

[74] Z. Tian et al. "FCOS: Fully Convolutional One-Stage Object Detection". In: *IEEE International Conference on Computer Vision*. 2019, pp. 9626–9635.

[75] Jonathan Tompson et al. "Efficient object localization using convolutional networks". In: *IEEE conference on computer vision and pattern recognition*. 2015, pp. 648–656.

[76] Ashish Vaswani et al. "Attention is all you need". In: *arXiv preprint arXiv:1706.03762* (2017).

[77] Neal E. Vlasic Bill; Boudette. "Self-Driving Tesla Was Involved in Fatal Crash, U.S. Says". In: *The New York Times* (2018). URL: https://www.nytimes.com/2016/07/01/business/self-driving-tesla-fatal-crash-investigation.html.

[78] Brianna Volz. "Family calls for investigation after Tesla driver dies in Osceola crash". In: *Olickorlando* (2019). URL: https://www.clickorlando.com/news/2019/09/26/family-calls-for-investigation-after-tesla-driver-dies-in-osceola-crash/.

[79] Chien-Yao Wang et al. "CSPNet: A new backbone that can enhance learning capability of CNN". In: *IEEE/CVF conference on computer vision and pattern recognition workshops*. 2020, pp. 390–391.

[80] Xinlong Wang et al. "Solo: Segmenting objects by locations". In: *European Conference on Computer Vision*. Springer. 2020, pp. 649–665.

[81] Graham V. Weinberg. "An Introduction to Radar Sliding Window Detectors". In: *arXiv preprint arXiv:1709.09786* (2017).

[82] M Williams. "PROMETHEUS-The European research programme for optimising the road transport system in Europe". In: *Driver Information, IEE Colloquium on*. IET. 1988.

[83] Sanghyun Woo et al. "Cbam: Convolutional block attention module". In: *European conference on computer vision (ECCV)*. 2018, pp. 3–19.

[84] Yuxin Wu et al. *Detectron2*. https://github.com/facebookresearch/detectron2. 2019.

[85] Dan Yadron Danny; Tynan. "Tesla driver dies in first fatal crash while using autopilot mode". In: *The Guardian* (2018). URL: https://www.theguardian.com/technology/2016/jun/30/tesla-autopilot-death-self-driving-car-elon-musk.

[86] Wojciech Zaremba, Ilya Sutskever, and Oriol Vinyals. "Recurrent neural network regularization". In: *arXiv preprint arXiv:1409.2329* (2014).

[87] Guoqiang Zhang, Haopeng Li, and Fabian Wenger. "Object detection and 3d estimation via an fmcw radar using a fully convolutional network". In: *IEEE International Conference on Acoustics, Speech and Signal Processing*. IEEE. 2020, pp. 4487–4491.

[88] H. Zhang et al. "Self-attention generative adversarial networks". In: *International Conference on Machine Learning*. PMLR. 2019.

[89] Shifeng Zhang et al. "Bridging the gap between anchor-based and anchor-free detection via adaptive training sample selection". In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, pp. 9759–9768.

[90] Zhong-Qiu Zhao et al. "Object detection with deep learning: A review". In: *IEEE transactions on neural networks and learning systems* 30.11 (2019), pp. 3212–3232.

[91] Barret Zoph and Quoc V Le. "Neural architecture search with reinforcement learning". In: *arXiv preprint arXiv:1611.01578* (2016).

•