uOttawa

Compact Object Detection with MetaFormers from Millimeter-Wave FMCW Radar Signals

Huaiyu Chen

Supervisor
Dr. Martin Bouchard
Dr. Robert Laganière

A thesis submitted to the University of Ottawa
In partial fulfillment of the requirements for the degree of
Master of Applied Science in
Electrical and Computer Engineering

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

© Huaiyu Chen, Ottawa, Canada, 2026

Abstract

With the development of sensor technologies, frequency-modulated continuous wave radars have seen increasing applications in the automotive industry. Operating on the millimeter-wave frequency bands, these radars offer stronger penetration capabilities through adverse weather conditions than optical sensors such as cameras and lidars, and provide accurate range and velocity measurements, making them a suitable choice for object detection tasks in autonomous driving systems. These embedded systems possess stringent requirements on the compactness and computational efficiency of the employed object detection models, which have not been adequately addressed by existing works in the literature. Motivated by the recent advancements in vision Transformers and the MetaFormer architecture, this thesis explores the design of well performing yet compact radar object detection models based on these novel architectures. A U-net shaped 3D Swin Transformer model is first developed to effectively capture spatio-temporal radar features for improved detection performance. Quantitative and qualitative analysis on the CRUW ROD2021 dataset demonstrate the effectiveness of the proposed architecture in modeling radar features and aggregating temporal information. Building upon this, this thesis proposes mRadNet, a more compact radar object detection model inspired by the MetaFormer architecture. In addition to the flexibility of leveraging both convolution-based and attention-based token mixers, mRadNet incorporates efficient token encoding and decoding strategies to further reduce model size and inference time, improving state-of-the-art performance on the CRUW ROD2021 dataset by a considerable margin with 20% smaller model size and 60% lower inference latency.

Acknowledgments

I would like to express my sincere gratitude to all those who have supported and guided me throughout the completion of this master's thesis. First and foremost, I am deeply grateful to my supervisors, Dr. Bouchard and Dr. Laganière, for their invaluable guidance, patience, and encouragement. Their insightful suggestions and rigorous academic standards have been essential to the development and completion of this research. I have benefited greatly from their expertise and dedication. I would also like to thank the faculty members of the School of Electrical Engineering and Computer Science for providing a rigorous academic environment and valuable courses that laid a solid foundation for this study. My appreciation also extends to my friends, whose discussions and companionship made this academic journey both enriching and enjoyable. Finally, I would like to express my gratitude to my family for their unconditional support, understanding, and encouragement throughout my studies. This thesis would not have been possible without their love and support.

Contents

List of Tables

List of Figures

List of Acronyms

ADAS Advanced Driver-Assistance System

ADC Analog-to-Digital Converter

AoA Angle of Arrival

AUC Area Under Curve

AP Average Precision

AR Average Recall

BEV Bird's Eye View

CNN Convolutional Neural Network

CFAR Constant False Alarm Rate

CUT Cell Under Test

CW Continuous Wave

FCN Fully Connected Network

FFN Feed-Forward Network

FFT Fast Fourier Transform

FMCW Frequency-Modulated Continuous-Wave

FN False Negative

FP False Positive

FPN Feature Pyramid Network

IF Intermediate Frequency

IoU Intersection over Union

L-NMS Location-based Non-Maximum Suppression

MAC Multiply-Accumulate

mAP mean Average Precision

mAR mean Average Recall

MHA Multi-Head Attention

MHSA Multi-Head Self-Attention

MLP Multi-Layer Perceptron

MSE Mean Squared Error

NLP Natural Language Processing

NMS Non-Maximum Suppression

NN Neural Network

OLS Object Localization Score

RAD Range-Azimuth-Doppler

RA Range-Azimuth

RD Range-Doppler

RNN Recurrent Neural Network

RoI Region of Interest

RPN Region Proposal Network

SISO Single Input Single Output

SOTA State-Of-The-Art

SDPA Scaled Dot Product Attention

SW-MSA Shifted Window Multi-Head Self-Attention

ToF Time of Flight

TN True Negative

TP True Positive

W-MSA Window Multi-Head Self-Attention

Chapter 1

Introduction

1.1 Introduction

Human errors are the leading cause of road accidents. In 2023, at least 62.9% of fatal collisions in Canada were attributed to human error [1], as shown in Figure 1.1. Advanced Driver-Assistance System (ADAS) technologies have been developed to assist drivers in perceiving their surroundings and making driving decisions, thereby minimizing human errors and improving traffic efficiency and safety. A critical component of ADAS, *Object Detection*, is the ability to accurately detect and localize road users and static objects in the vehicle's vicinity. It acts as the foundation for subsequent tasks including trajectory prediction and

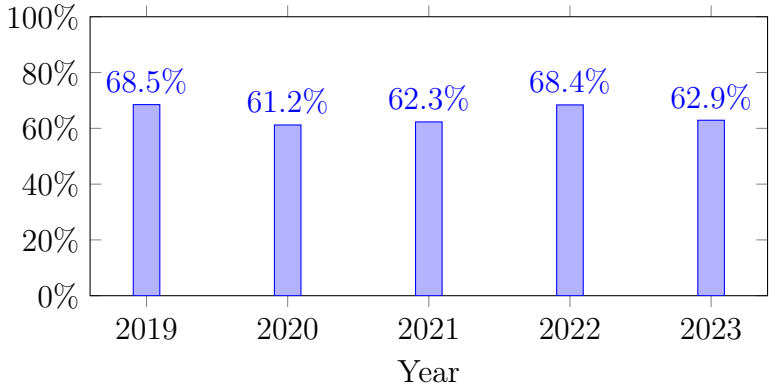


Figure 1.1: Human error in fatal collisions. In Canada, human errors account for more than 60% of fatal collisions every year from 2019 to 2023.

motion planning, and remains as one of the most challenging tasks in ADAS development.

With recent advancements in deep learning, significant progress has been made in object detection using camera images. More efficient and accurate models have been proposed, and the computational power of embedded systems has largely increased, making it feasible for image-based object detection models to be applied in real-world applications. However, image-based object detection still faces challenges in adverse weather and lighting conditions, such as heavy rain, fog, and nighttime driving, their performance degrades significantly when there is limited visibility since cameras heavily rely on ambient light to capture visual information.

To address the limitations of image-based object detection, emission-based sensors such as lidar have been increasingly adopted in ADAS systems. Lidar sensors emit scanning laser beams and sample the reflection times in different directions to generate a 3D point cloud representation of the surrounding environment, in which each point represents a reflection from an object's surface. Lidar sensors can operate effectively in low-light conditions since they do not rely on ambient light. They also provide accurate depth information, which is crucial for object localization. However, being an optical sensor, lidar suffers from some of the same limitations as cameras in adverse weather conditions, such as heavy rain and fog, where the laser beams can be scattered or absorbed by water droplets. Faulty sample points (known as *ghosts*) can appear in reflective environments such as near glass buildings or wet roads, leading to misinterpretation of the scene.

Radar sensors, on the other hand, operate at a much higher wavelength (typically in the order of millimeters) than lidar and cameras and are therefore less affected by adverse weather conditions. Radar signals also carry information about the speed of objects, which is not available in lidar and camera data. While radar sensors have traditionally been used in low resolution applications such as adaptive cruise control, recent advancements in Frequency-Modulated Continuous-Wave (FMCW) radar technology have enabled the development of high-resolution radar sensors, paving the way for their application in object detection tasks.

Still, radar object detection remains a challenging task due to radar signals being susceptible to noise and the lack of texture information, in addition to the obscure nature of radar signals. This puts forth the need for stronger object detection models that can effectively leverage the unique characteristics of radar signals, while also being efficient enough to be deployed in real-time applications.

In this thesis, a spatio-temporal radar object detection model is proposed based on 3D Swin Transformers [2]. We demonstrate Swin Transformers' effectiveness in modeling temporal features and compensating for the noise and sparsity of radar signals. Furthermore, we design a compact radar object detection model based on 3D MetaFormers [3], coined *mRadNet*, to improve the performance of the model proposed in chapter 4. Benefitting from the MetaFormer architecture, mRadNet achieves more accurate detection with a fraction of the model size and inference time compared to Transformer-based models, making it suitable for real-time applications. The proposed models are evaluated on the CRUW ROD2021 dataset [4, 5], where mRadNet surpasses the state-of-the-art performance in terms of accuracy, model size, and inference speed.

1.2 Motivation

Radar has seen increasing adoption in ADAS systems as a supplement to cameras and lidar, but single-sensor radar object detection remains a challenging task due to reasons discussed in section 1.1. Existing single-frame radar object detection models often struggle with the noise of radar signals, leading to false positives and missed detections. These models also require large model sizes to achieve acceptable performance, which leads to high inference latency and limits their deployment in embedded systems. Motivated by this, we aim to develop a radar object detection model that is less susceptible to signal noise, while maintaining a small model size. Input / output latency and inference latency is also taken into consideration, as it is crucial for the safety and reliability of ADAS systems.

1.3 Thesis Contributions

Our contributions are as follows:

- We show that by leveraging temporal features, Swin Transformers can effectively model the spatio-temporal characteristics of radar signals, leading to improved object detection performance.
- We propose a MetaFormer-based radar object detection model, mRadNet, that achieves state-of-the-art performance on the CRUW ROD2021 dataset with a significantly smaller model size and lower inference latency compared to existing Transformer-based models.
- We introduce more efficient token encoding and decoding strategies in mRadNet to facilitate the effectiveness of token mixers and the model's compactness.
- We compose a conference paper [6] based on the work presented in chapter 5, and release the source code for public access: huaiyu-chen/mRadNet.

1.4 Thesis Outline

The remainder of this thesis is organized as follows:

- chapter 2 provides background information on relevant topics. Fundamental concepts and techniques in FMCW radar signal processing, deep learning and object detection are introduced.
- chapter 3 provides a review of literature in relevant areas including object detection using images and radar signals.
- chapter 4 proposes a spatio-temporal radar object detection model based on 3D Swin Transformers. The backbone architecture is described, followed by the addition of

temporal modeling capabilities through 3D Attention layers, and the design of the output space. Experiments are conducted to evaluate the performance of the proposed model. Results are analyzed via case studies and visualizations to reveal the strengths and weaknesses of the model.

- chapter 5 designs a compact radar object detection model aiming to improve the performance of the previous model and address issues found in chapter 4. The overall architecture is described, and token encoding and decoding strategies are presented, along with the choice of token mixers. Experiments are conducted to evaluate the performance of the proposed model, and comparisons are made with the model proposed in chapter 4 and other state-of-the-art models to demonstrate the effectiveness and efficiency of the proposed model.

- chapter 6 concludes the thesis and suggests potential directions for future work.

Chapter 2

Background

This chapter provides the necessary background information on the topics relevant to this thesis. It covers the principles of FMCW radar, deep learning, and object detection, which are essential for understanding the subsequent chapters.

2.1 FMCW Radar

2.1.1 Radar Signal Processing

Radar, acronym for **ra**dio **d**etection **a**nd **r**anging, have seen long-established use in various applications, from military to civilian. As its name suggests, radars emit radio waves to detect objects and measure their distance with the echo signal. Depending on the emitted signal, radars can be classified into different types, as shown in Figure 2.1 [7].

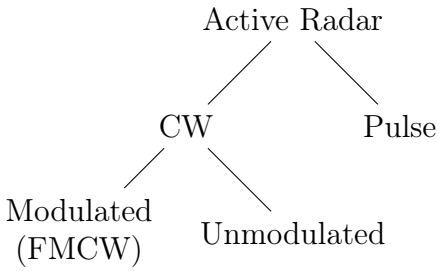


Figure 2.1: **Types of radar systems.** Passive radars are not shown here.

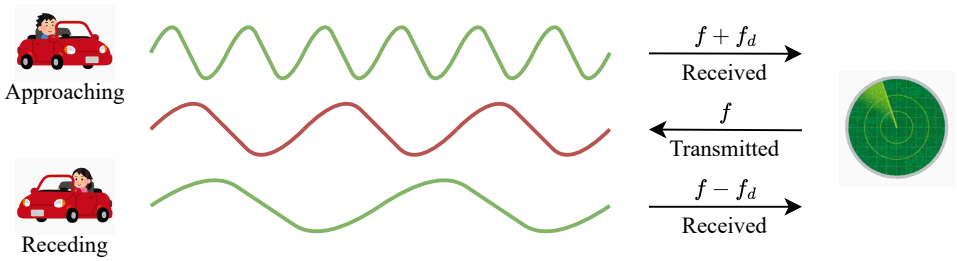


Figure 2.2: Doppler effect in continuous wave radar. The frequency of the received signal is shifted based on the relative velocity of the target.

Pulse Radar

Pulse radars emit short bursts of radio waves and measure the Time of Flight (ToF), or the time it takes for the signal to travel to the target and back. Given that radio waves travel at the speed of light, the distance to the target can be calculated as shown in Equation 2.1.

$$d = \frac{c \cdot \Delta t}{2} \tag{2.1}$$

where d is the distance to the target, c is the speed of light, and Δt is the ToF. Pulse radars are typically Single Input Single Output (SISO) sensors, using only one set of transmitter and receiver, which means that the radar can only measure the distance to one target at a time. In aviation and maritime applications, pulse radars overcome this limitation by rotating the antenna, allowing the radar to scan a wide area and detect multiple targets sequentially, and the velocity of the targets can be estimated based on the displacement of the targets between scans. However, this scheme is not suitable for applications that require real-time tracking of multiple targets, such as in automotive applications.

Continuous Wave Radar

Continuous Wave (CW) radars, on the other hand, emit a continuous wave signal to detect objects. Unmodulated CW radars measure the frequency shift of the received signal caused by Doppler effect to detect the velocity of moving objects, as shown in Figure 2.2. The

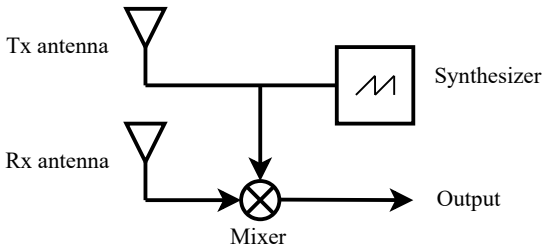


Figure 2.3: Diagram of an SISO FMCW radar system. The received signal is mixed with the transmitted signal to generate the IF signal.

Doppler shift is given by Equation 2.2.

$$f_d = 2v\frac{f}{c} \tag{2.2}$$

where f_d is the Doppler frequency shift, v is the relative velocity of the target, f is the frequency of the transmitted signal, and c is the speed of light. Unmodulated CW radars are also called Doppler radars due to their reliance on the Doppler effect. They can measure the velocity of moving objects, but they cannot measure the distance to the target, as the received signal is continuous and does not provide information about the ToF. To overcome this limitation, FMCW radars were developed.

FMCW Radar

FMCW radars are CW radars with frequency modulated signals, which means the frequency of the transmitted signal changes over a period of time known as a *chirp*. Different modulation schemes can be used to generate the chirp, such as sine wave, sawtooth wave, triangular wave, square wave, etc., with the most common being sawtooth wave and triangular wave. For the sake of illustration, we only consider the sawtooth wave due to its simplicity. A group of consecutive chirps is called a *frame*.

In FMCW radars, a device called a *mixer* is used to mix the received signal with the transmitted signal, generating an Intermediate Frequency (IF) signal, as shown in Figure 2.3 [8]. For a transmitted signal with frequency f_T and phase ϕ_T, the received signal with

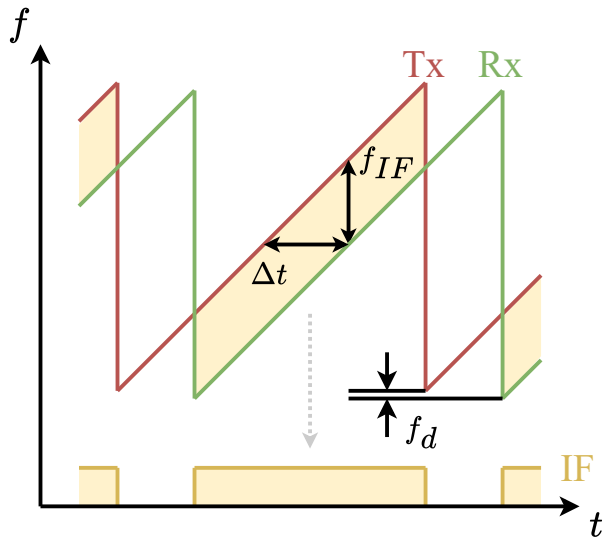


Figure 2.4: Chirp signals in FMCW radar systems. The transmitted signal is mixed with the received signal to generate the IF signal.

frequency f_R and phase ϕ_R, the IF signal is given by Equation 2.3.

$$\begin{aligned} f_{IF} &= f_T - f_R \\ \phi_{IF} &= \phi_T - \phi_R \end{aligned} \tag{2.3}$$

where f_{IF} is the frequency of the IF signal, and ϕ_{IF} is the phase of the IF signal. Given that the received signal can be seen as a delayed version of the transmitted signal, the IF signal should have constant frequency, as shown in Figure 2.4. Note that the Doppler shift f_d is not considered here, as it is negligible compared to the frequency of the transmitted signal.

FMCW radars emit radio waves in a continuous manner, so the ToF cannot be measured directly as in pulse radars. Instead, the ToF is estimated based on the frequency of the IF signal and the slope of the chirp signal. The ToF is given by Equation 2.4.

$$\Delta t = \frac{f_{IF}}{S} \tag{2.4}$$

where Δt is the ToF, f_{IF} is the IF signal frequency, and S is the slope of the chirp signal. If we

substitute Δt in Equation 2.1, we get the distance to the target R as shown in Equation 2.5.

$$R = \frac{c \cdot f_{IF}}{2S} \tag{2.5}$$

Note that in a known FMCW radar system, the distance to a target R is only dependent on the frequency of the IF signal f_{IF}. When multiple targets at different distances are present, the IF signal is a superposition of multiple sinusoidal signals with different frequencies. Therefore, each target is represented by a peak at a specific frequency in the IF signal's spectrum.

The Doppler effect is used in CW radars to measure the velocity of moving objects. In FMCW radars, however, the Doppler shift is not significant enough compared to the frequency of the transmitted signal. Consider a 77 GHz millimeter wave radar with a chirp time $T_c = 40\mu s$ and a slope $S = 50\text{MHz}/\mu s$, an object moving at $v = 20\text{m/s}$ produces a Doppler shift $f_d = 2v\frac{f}{c} = 513\text{Hz}$ and a change in IF signal frequency of $\Delta f_{IF} = \frac{2ST_cv}{c} = 267\text{Hz}$, which are not discernible in the IF signal spectrum. The phase of the IF signal, however, is very sensitive to small target displacements between chirps.

Recall that phase of the IF signal is the difference between the phase of the transmitted signal and the phase of the received signal, given by Equation 2.3. When the object moves at speed v, the phase of the transmitted signal is invariant, and the phase change of the received signal is given by Equation 2.6.

$$\begin{aligned} \Delta\phi_R &= 2\pi\frac{2\Delta R}{\lambda} = \frac{4\pi v T_c}{\lambda} \\ v &= \frac{\lambda\Delta\phi_R}{4\pi T_c} \end{aligned} \tag{2.6}$$

where $\Delta\phi_R$ is the phase change of the received signal, ΔR is the displacement of the target, and λ is the wavelength of the transmitted signal. Going back to the aforementioned example, the phase change of the received signal is $\Delta\phi_R = 2.58 = 148°$.

Similar to distance measurement, the velocity of a target can be represented by a peak

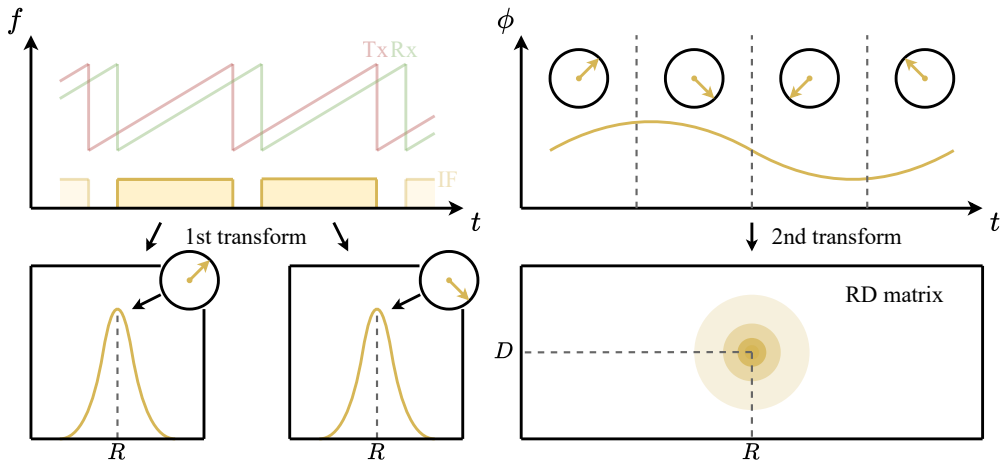


Figure 2.5: The 2D Fourier transform process. The first transformation is performed on the IF signals to acquire the target distance R. The IF spectra have peaks at the same frequency (insensitive to target distance), but the phase is different. The second transformation is performed on the phases from different chirps to acquire the target velocity D.

on the frequency domain, as shown in Figure 2.5 [9]. Given that the velocity of a target is proportional to the derivative of the phase of the IF signal, a Fourier transform can be applied to the spectra of the IF signals from different chirps. The result is a two-dimensional matrix, where the first dimension represents the frequency of the IF signal, and the second dimension represents the frequency of the phase change. A peak in this matrix indicates the presence of a target at a specific distance and velocity. This matrix is called the Range-Doppler (RD) matrix.

The above text describes the basic working principle of a SISO FMCW radar system. It is capable of measuring the target distance and velocity, but the direction of the target, or Angle of Arrival (AoA), cannot be measured. Recall that small displacements of the target cause a phase change of the IF signal. With multiple receiving antennas, the AoA can be estimated based on the distance difference from the antennas to the target, as shown in Figure 2.6.

AoA estimation requires at least 2 receiving antennas arranged in a horizontal array. Received signals from different directions introduce a path length difference for the antennas,

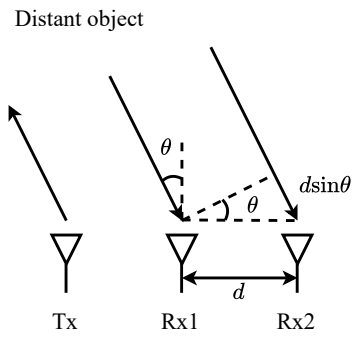


Figure 2.6: AoA estimation in multiple input FMCW radar systems. Each antenna in the array measures a different distance to the target, which corresponds to a different phase of the IF signal. These radar systems are also known as 3D radars. Note that distance between antennas is negligible compared to the target distance.

which results in a phase difference between the IF signals, given by Equation 2.7.

$$\begin{aligned} \Delta\phi_R &= \frac{2\pi d \sin\theta}{\lambda} \\ \theta &= \arcsin\left(\frac{\Delta\phi_R \lambda}{2\pi d}\right) \end{aligned} \tag{2.7}$$

where d is the distance between the antennas, θ is the AoA of the target, and λ is the wavelength of the transmitted signal. In order to obtain the maximum AoA measurement range of $\pm\frac{1}{2}\pi$, the distance between the antennas must be less than half of the wavelength of the transmitted signal, i.e. $d < \frac{\lambda}{2}$, to avoid phase ambiguity. Note that the pitch angle of the target cannot be measured with horizontal arrays. In order to obtain a full 3D representation of the target and to increase the resolution, a 2D array with more antennas is required to form a 4D radar system (i.e., range, speed (Doppler), azimuth angle, and elevation angle), but for the sake of this thesis we only consider 3D radars (i.e., range, speed (Doppler), and azimuth angle). Similar to velocity measurement, the AoA of the target can also be acquired with a Fourier transform, either on the raw IF signals or on the RD matrice. This process will be discussed in more detail in subsection 2.1.2.

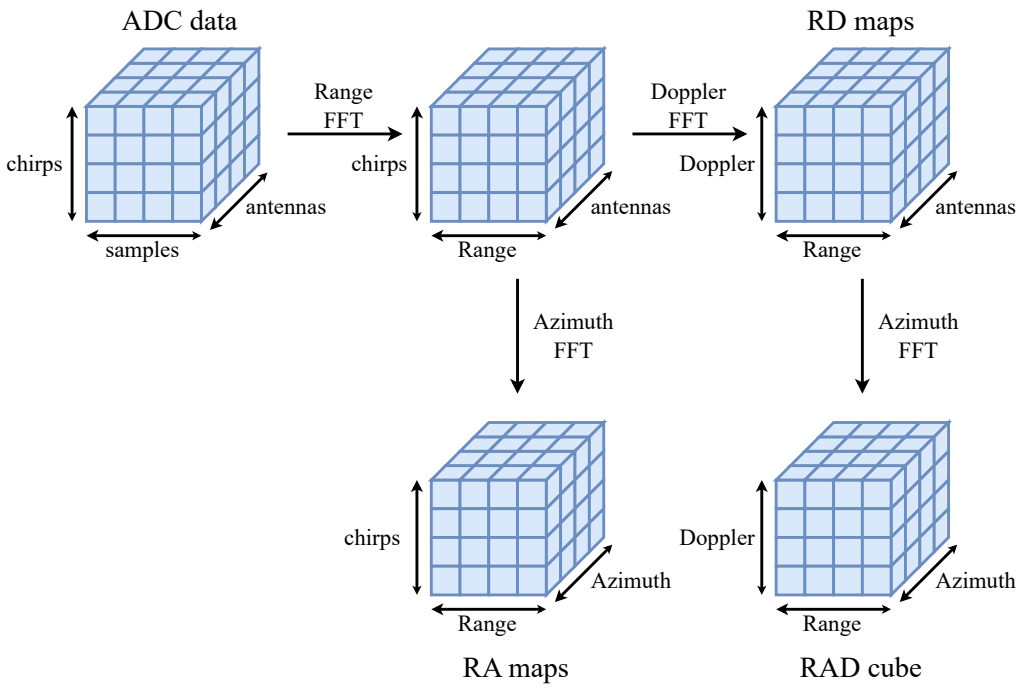


Figure 2.7: Radar data representations. Three FFTs are applied to the three axes of the ADC matrix to produce the RAD cube.

2.1.2 Radar Data Representations

In subsection 2.1.1, we showed that a multiple input FMCW radar system can measure the distance, velocity and AoA of targets. These measurements are estimated from the IF signals from each antenna, which contains a fixed number of chirps sampled by an Analog-to-Digital Converter (ADC) and converted into a digital signal. In deep learning applications, the raw ADC signals are arranged into a 3D matrix, with its three dimensions representing the number of samples in each chirp, the number of chirps in a frame, and the number of antennas in the array, as shown in Figure 2.7.

Recall that the three measurements, distance, velocity, and AoA, can be represented by peaks in the frequency domain. Therefore, Fast Fourier Transforms (FFTs) can be applied to each axis of the ADC matrix to extract these measurements. The Range-FFT is applied to the samples axis to extract the distance information, the Doppler-FFT is applied to the chirps axis to extract the velocity information, and the Azimuth-FFT is applied to the

antennas axis to extract the AoA information. The result is a 3D matrix called the Range-Azimuth-Doppler (RAD) cube, where each point in the cube represents the response at a specific distance, velocity, and AoA. Note that it is not always required to apply all three FFTs to the ADC matrix, as some applications may not require all three measurements, and deep learning models can also work with data in the time domain. Apart from the RAD cube, use of Range-Azimuth (RA) maps (from Range-FFT and Azimuth-FFT), as well as RD maps (from Range-FFT and Doppler-FFT) are also commonly seen in works from the literature. Some works have also explored the possibility of extracting information directly from raw ADC data with deep learning models.

Following the success of lidar object detection, the point cloud representation of radar data has also been explored in the literature. A detector (usually based on thresholding) is applied to the RAD cube to determine the presence or absence of targets. Points with high response values are extracted, while low response values are discarded. The result is an array of detected points known as a point cloud, where each point is associated with attributes such as intensity, velocity, etc. While this representation benefits from existing point cloud processing techniques, loss of information is inevitable, and the number and location of points across frames are largely affected by factors such as noise, even when the target is stationary. In this thesis, we will not consider point cloud representations, but rather focus on frequency domain representations mentioned above.

2.2 Deep Learning

Over the last decade, deep learning has emerged as a dominant paradigm in machine learning, revolutionizing various fields such as computer vision, natural language processing, and speech recognition. Laying its foundations on the human brain's neural architecture, deep learning leverages Neural Networks (NNs) to model statistical patterns in data, enabling machines to perform tasks on unseen data without explicit programming. NNs are com-

posed of nodes, or *neurons*, organized in layers, and connected by weighted edges. Each node takes the sum of weighted inputs (real value) from preceding nodes, and applies a non-linear activation function to produce an output. A simplest form of NN is described in Equation 2.8.

$$Z = A(WX + b) \tag{2.8}$$

where W and b are the weights and bias, X is the input, Z is the neuron's output, and $A(\cdot)$ is a non-linear function called *activation function*. The weights and the bias together form the model's parameters, which define the model's behavior. When the activation function is a step function, Equation 2.8 represents a *Perceptron*, the earliest form of NN. Introduced by Frank Rosenblatt in 1958 [10], the Perceptron laid the groundwork for more complex architectures, but it was limited to linearly separable problems. The limitations of the Perceptron led to the development of Multi-Layer Perceptron (MLP), which introduced hidden layers and non-linear activation functions, allowing for the modeling of more complex relationships in data.

The model's parameters are initialized randomly. For the model to learn from a known set of data, a process called *training* is performed, in which the model's parameters are optimized through *backpropagation* [11]. Backpropagation is an optimization method that computes the gradient of a loss function with respect to the model's parameters using the chain rule, to minimize the difference, or *loss*, between the predicted output and the expected output.

Consider the single layer NN in Equation 2.8. In each iteration, a random subset of the training data, or a *batch*, is passed through the network to generate the output, and a loss function L is used to measure the loss between the predicted output and the expected output. The gradient of the loss function with respect to the weights and bias are calculated

as shown in Equation 2.9.

$$\begin{aligned} \frac{\partial L}{\partial W} &= \frac{\partial L}{\partial A} \cdot \frac{\partial A}{\partial Z} \cdot \frac{\partial Z}{\partial W} = L' \cdot A' \cdot X \\ \frac{\partial L}{\partial b} &= \frac{\partial L}{\partial A} \cdot \frac{\partial A}{\partial Z} \cdot \frac{\partial Z}{\partial b} = L' \cdot A' \cdot 1 \end{aligned} \tag{2.9}$$

It is worth noting that the gradients depend on the derivatives of the activation function A and the loss function L, requiring these functions to be differentiable globally. The design of these functions is critical for the model's performance, since their gradients are then used to update the model's parameters in the opposite direction of the gradient, scaled by a learning rate α, as shown in Equation 2.10.

$$\begin{aligned} W &\leftarrow W - \alpha \cdot \frac{\partial L}{\partial W} \\ b &\leftarrow b - \alpha \cdot \frac{\partial L}{\partial b} \end{aligned} \tag{2.10}$$

This process is repeated for multiple iterations, until the model converges to a set of parameters that minimize the loss function, effectively learning from the training data.

MLPs were also referred to as Fully Connected Networks (FCNs) due to their dense connection between layers. Early FCNs were limited to shallow architectures with small number of neurons, because their parameter size grows exponentially with the number of layers and neurons, making them computationally expensive and prone to overfitting, but these limitations would be largely addressed by the emergence of Convolutional Neural Networks (CNNs) [12].

2.2.1 Convolutional Neural Networks

CNN was inspired by a preceding work in neuroscience, where the visual cortex of animals was observed to contain neurons that respond to stimuli in a localized region of the visual field, known as the receptive field. As its name suggests, CNNs are built around the convolution operation. Equation 2.11 provides the mathematical definition for a 2D discrete convolution

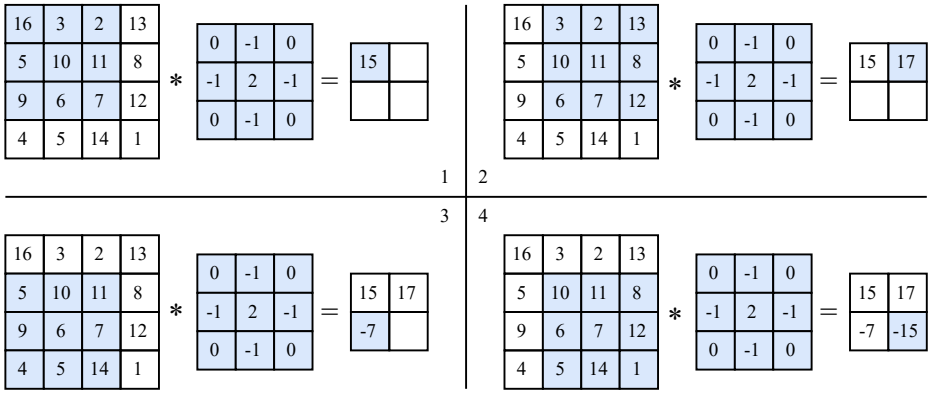


Figure 2.8: 2D Convolution. A 3×3 convolution kernel is applied to a 4×4 input matrix, producing a 2×2 output matrix. In each step, the kernel slides over the input matrix, and computes the weighted sum of the input values within the kernel's receptive field.

operation.

$$(f * g)[x, y] = \sum_{i=-\infty}^{\infty} \sum_{j=-\infty}^{\infty} f[i, j] g[x - i, y - j] \tag{2.11}$$

where f and g are two 2D discrete functions. In each layer, a convolution kernel with learnable parameters is convolved over the input data, computing a weighted sum of the input values within the kernel's *receptive field*, as shown in Figure 2.8. The same kernel is reused across the entire input, based on the prior knowledge that images are shift invariant. Such convolution layers can be cascaded to form a hierarchical structure, where the effective receptive field of a kernel expands as information propagates through layers, allowing the model to learn increasingly complex features from the input data.

Since its introduction, CNNs have become the de facto architecture for computer vision tasks. The success of CNNs can be attributed to their scalable and lightweight architecture, allowing them to learn hierarchical features from high-dimensional data, while significantly reducing the number of parameters compared to FCNs.

2.2.2 Transformers

While CNNs have dominated computer vision tasks, a new architecture called *Transformers* has emerged as a powerful alternative [13]. Transformers were first introduced in the con-

text of Natural Language Processing (NLP) to address the limitations of Recurrent Neural Networks (RNNs) in modeling long-range dependencies in sequential data and parallelizing training, but their success has led to their adoption in various domains, including computer vision.

Transformers are built around the *attention* mechanism. Inspired by the human cognitive process of focusing on relevant information, the attention mechanism allows the model to weigh the importance of different input elements on a global scale, rather than relying on local receptive fields as in CNNs. The Scaled Dot Product Attention (SDPA) used in the original Transformer architecture is given by Equation 2.12.

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{2.12}$$

where Q, K, and V are inputs known as *query*, *key*, and *value* respectively, usually acquired from features with learnable linear projections, and d_k is the dimension of the key vector. By computing the dot product between Q and K, an attention score is obtained, which is then normalized using the softmax function to produce a probability distribution. This distribution is then used as weights for the values V, allowing the model to focus on relevant information from the input. When Q, K, and V are derived from the same input, the attention mechanism is referred to as *self-attention*, otherwise it is referred to as *cross-attention*.

Despite its ability of global dependence modeling, the above attention mechanism is not without limitations. It is prone to overfit unwanted patterns in the input, e.g. a token's relation with themselves or with special tokens added to facilitate training. To alleviate this issue, a technique called Multi-Head Attention (MHA) is commonly used, where multiple attention mechanisms, known as *heads*, are applied on different subspaces of the input, and

their outputs are concatenated, as shown in Equation 2.13.

$$\begin{aligned} \text{MultiHeadAttention}(Q, K, V) &= \text{Concat}\left(\text{head}_1, \ldots, \text{head}_h\right) W^O \\ \text{head}_i &= \text{Attention}(QW_i^Q, KW_i^K, VW_i^V) \end{aligned} \tag{2.13}$$

where W_i^Q, W_i^K, and W_i^V are learnable linear projections to transform the query, key, and value to their different subspaces, respectively. The outputs of all heads are concatenated and projected by a learnable matrix W^O to produce the final output. This technique can be seen as a form of ensemble learning, where each head is a weak learner that focuses on different subspaces of the input, and their outputs are combined to form a stronger representation.

Transformers employ a hierarchical architecture, where the modules, or *Transformer blocks*, can be stacked to form a deep network. A typical form of a Transformer Block is shown in Figure 2.9.

A Transformer Block consists of two cascading components: an attention layer and a Feed-Forward Network (FFN), each of which takes a normalized input and learns the residual between the input and the output. This approach, known as *residual learning*, was first introduced by ResNet. It allows the model to focus on the difference between the input and output, and helps to alleviate the vanishing gradient problem in deep networks.

The attention layer is used to model the global dependencies in the input. At a high level, it allows information exchange between different tokens by "mixing the tokens", effectively acting as a token mixer. However, the attention layer does not provide non-linearity, since its output is the linear product of the input and a weight matrix. Thus the FFN is used to introduce non-linearity and prevent the model from collapsing to a linear function.

Vision Transformer (ViT) [14] is a notable example of Transformer-based architectures in computer vision. It treats images as sequences of tokens, similar to how Transformers process text. Each token is concatenated with a learnable positional embedding to retain spatial information, and the resulting sequence is fed into an encoder consisting of multiple Transformer blocks. The output of the encoder is then used for various tasks, such as image

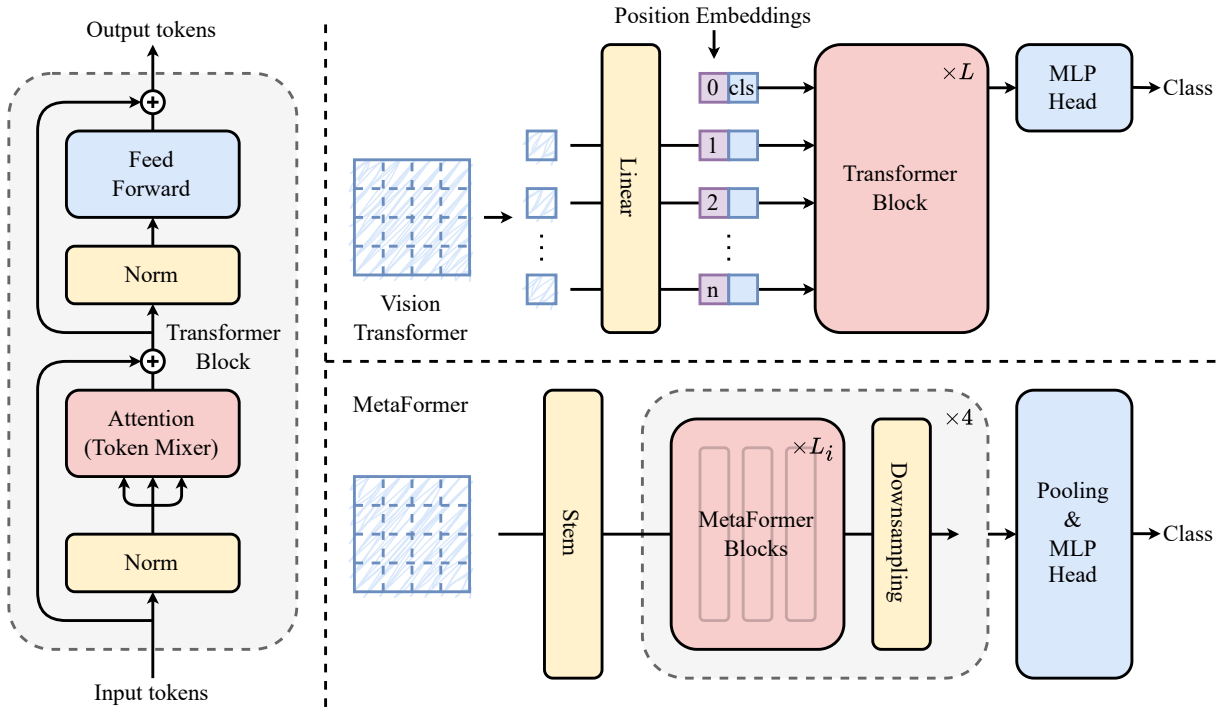


Figure 2.9: Transformer architectures. The left side shows the structure of a Transformer block. Note that the block shown here uses a *Pre-Norm* configuration commonly used by modern Transformer architectures, instead of the original *Post-Norm* configuration. The right side shows two examples of Transformer-based architectures in computer vision: ViT and MetaFormer.

classification or object detection. ViT was the first to show that Transformers can be used as a general-purpose architecture for computer vision, and it paved the way for the development of other Transformer-based vision architectures.

MetaFormer [15, 3] on the other hand, is a more recent architecture that extends the Transformer concept to a more general form. It argues that the success of Transformers is mainly attributed to the overall architecture instead of the attention mechanism, and by replacing the token mixer with any other operation that can mix tokens, e.g. pooling, the model can still easily surpass various CNNs and Transformer-based architectures. The MetaFormer architecture will be discussed in more detail in section 5.1.

2.3 Object Detection

Over the past decades, object detection has remained a central problem in computer vision, with applications ranging from security surveillance to autonomous driving [16]. Object detection refers to the task of determining where objects are located in an image (object localization) and identifying what those objects are (object classification). Intuitively, object detection can be achieved by solving these two sub-problems: localization and classification [17].

The location of an object can be represented in various ways, such as coordinates of the object center, a box surrounding the object (a *bounding box*), or a segmentation mask that delineates the object shape. The center point representation describes the location of an object with an (x, y) coordinate, it is often used in tasks where the shape or the size of the object is not critical, such as in human pose estimation or object tracking. The bounding box representation describes the location of an object with a rectangle defined by its top-left corner (x_{min}, y_{min}) and bottom-right corner (x_{max}, y_{max}) coordinates, or alternatively by its center point (x, y) and width and height (w, h). This representation is commonly used in most object detection tasks. The segmentation mask representation is often categorized to a

more fine-grained task called *image segmentation*, which is closely related to object detection but not the main focus of this thesis.

The classification of an object involves assigning a label to the detected object, which can be a single class label (e.g., "car", "pedestrian", "bicycle") or multiple labels in the case of multi-label classification. Usually, a fixed set of classes needs to be defined, on which the model is trained and evaluated, but unseen classes can also be detected using techniques like few-shot learning or zero-shot learning. Traditionally, features of an object were extracted using handcrafted methods, and statistical classifiers were used to generate a hyperplane that separates the feature space into classes. However, with the advent of deep learning, neural networks have become the dominant approach for object detection, allowing for end-to-end learning of both feature extraction and classification. The outputs of neural network classifiers are usually a posterior probability distribution over the classes, or *confidences*.

When treating object detection as a combined task of localization and classification, the latter seems to have dependencies on the former. The most intuitive way of solving the problem is to follow a two-stage approach, by localizing the objects first and then classifying them. Each stage needs to extract their own features since localization and classification requires different features on different scales. With the introduction of neural networks, however, the ability of automatic feature extraction has enabled two-stage methods to share the same feature extraction network, significantly reducing the computational cost. Going further, one-stage methods generate a class probability distribution over the entire image instead of for each object, thus removing this dependency and parallelizing the two tasks to achieve even faster inference [16]. In section 3.1, we will discuss the two-stage and one-stage object detection methods in more detail, including their architectures and how they handle the localization and classification tasks.

Chapter 3

Related Works

This chapter provides an overview of related works in the field of object detection, focusing on both image-based and radar-based approaches. Representative methods and their contributions to the field are discussed, highlighting the evolution of techniques and the integration of radar data with image-based object detection.

3.1 Object Detection using Images

Recent advancements in deep learning have brought significant improvements to object detection in robustness, accuracy and speed. Deep learning-based object detection methods can be broadly categorized into two main approaches: two-stage detectors and one-stage detectors. In this section, we will explore the ideologies behind these approaches, their architectures, and the key models that have shaped the field.

3.1.1 Two-Stage Methods

After AlexNet [18] brought the field of computer vision into the CNN era, the first significant advancement in object detection was the introduction of two-stage methods. These methods try to split the object detection problem into two sub problems: object localization and classification. As shown in Figure 3.1 (a), two-stage methods typically consist of two

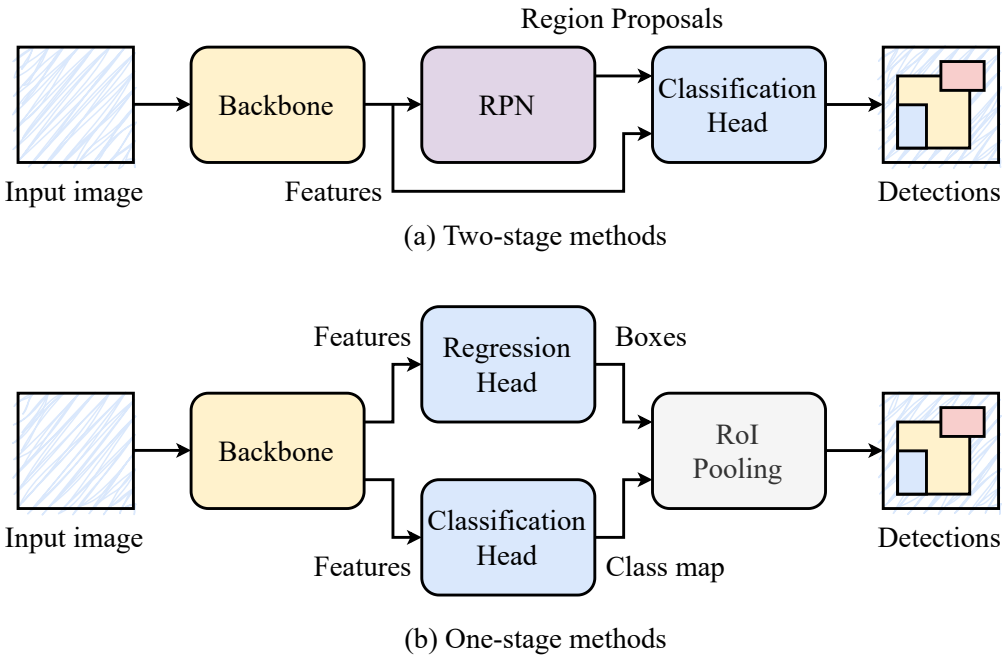


Figure 3.1: Comparison of one-stage and two-stage object detection methods. One-stage methods predict bounding boxes and class probabilities directly from the feature map, while two-stage methods first generate region proposals and then classify them.

main stages: the first stage (usually a Region Proposal Network (RPN)) generates a set of candidate object proposals, and the second stage (the classification head) classifies these proposals into specific object categories. By framing the detection as a "coarse-to-fine" process, two-stage methods focus on Regions of Interest (RoIs) with high potential for containing objects, allowing for more accurate classification and localization in complex scenes, but at the cost of increased computational complexity and inference time.

R-CNN (Region Based Convolutional Neural Network) [19] was one of the first models to demonstrate the effectiveness of deep learning in object detection, and also paved the way for subsequent two-stage methods. R-CNN follows a simple yet effective pipeline: it first uses a selective search algorithm to generate around 2000 region proposals from the input image, then extracts features from these regions using a pre-trained CNN (e.g., AlexNet), and finally classifies each region using a set of support vector machines (SVMs). R-CNN also employs bounding box regression to refine the localization of the detected objects. However, R-CNN is computationally expensive due to the need to run the CNN on each proposal separately. The

feature extractor and the classifier also need to be trained separately, rendering it inefficient for large datasets.

Fast R-CNN [20] improves upon R-CNN by sharing the computation of the CNN across all region proposals. Instead of cropping the regions and running the CNN on each one, Fast R-CNN feeds the entire image through the CNN once to obtain a feature map. It then uses a RoI pooling layer to extract fixed-size feature vectors for each proposal from the feature map. This significantly speeds up the detection process and allows for end-to-end training of the model. Fast R-CNN also introduces a multi-task loss function that jointly optimizes classification and bounding box regression. With these improvements, Fast R-CNN achieves state-of-the-art performance with faster training and inference times, and the bottleneck falls on the region proposal generation step.

Faster R-CNN [21] takes a further step by introducing a RPN that shares the convolutional features with the detection network. The RPN is a small network that predicts objectness scores and bounding box coordinates for a set of anchor boxes at each spatial location in the feature map. The RPN generates region proposals directly from the feature map, eliminating the need for an external proposal generation step like selective search. This allows Faster R-CNN to achieve real-time performance while maintaining high accuracy.

3.1.2 One-Stage Methods

YOLO (You Only Look Once) [22] emerged as the first one-stage object detection framework. As a contemporary of Fast R-CNN, YOLO tries to remove R-CNN's bottle neck from another perspective. Instead of using a dedicated algorithm or network for bounding box candidate generation, YOLO starts from a set of predefined center points, and predicts the candidates' size and displacement from these centers. Specifically, the input image is first divided into $S \times S$ grid cells. For each cell, the network generates B bounding boxes, each represented by a vector of $(x, y, w, h, confidence)$, where x and y are displacements from the block center, and w and h are the bounding box size. The network also needs to classify each block into

one of C categories. The final output is a $S \times S \times (B \times 5 + C)$ tensor. Not only does this approach eliminate the need for a separate proposal generation step, but it also allows for end-to-end training of the entire network. YOLO's architecture is designed to be fast and efficient, making it suitable for real-time applications. However, YOLO's grid-based approach can struggle with small objects and overlapping bounding boxes, as it may assign multiple objects to the same grid cell.

Over the years, YOLO has undergone numerous iterations, with each version improving upon the previous one in terms of accuracy and speed. YOLOv2 [23] introduced anchor boxes to address the issue of bounding box regression, allowing the model to predict multiple bounding boxes per grid cell. YOLOv3 [24] further improved the architecture by integrating Feature Pyramid Network (FPN) [25] to detect objects at different scales, and introduced a multi-label classification approach to handle overlapping classes. Variants of YOLO have continued to achieve state-of-the-art performance in object detection benchmarks, and have since been widely adopted in various applications.

In recent years, Transformers have deeply influenced the field of computer vision, including object detection. DETR (DEtection TRansformer) [26] is a pioneering one-stage model that applies the Transformer architecture to object detection tasks. DETR uses a conventional CNN backbone for preliminary feature extraction, and then applies a Transformer encoder-decoder architecture to predict a fixed number of object bounding boxes and class labels. The Transformer decoder takes the encoded features and a set of learnable object queries as input, and outputs a set of predictions that either correspond to an object or a "no object" class. A novel bipartite matching loss is used to match the predicted boxes with the ground truth boxes. By treating object detection as a direct set prediction problem, DETR eliminates the need for multiple components in previous models that encode prior knowledge, including anchors and Non-Maximum Suppression (NMS). This approach enables end to end training of the entire model, and significantly reduces the computational complexity of the detection process, However, it also results in an increased training time

due to the absence of semantic guidance from anchors, as well as the difficulty in learning the object queries. DETR also struggles to detect small objects since the feature maps produced by the CNN backbone are heavily downsampled. Nevertheless, DETR has demonstrated impressive performance on various object detection benchmarks, and has since inspired a new wave of one-stage object detection Transformers such as Deformable DETR [27] and RT-DETR [28].

3.2 Object Detection using FMCW Radar Signals

3.2.1 Datasets

Dataset	Year	Radar type	Data format	Labels	Size
CARRADA [29]	2020	FMCW	RA, RD, RAD	2D boxes	13k frames
RADIATE [30]	2020	Pulse	RA*	2D boxes	44k frames
CRUW (ROD2021) [4, 5]	2021	FMCW	RA	2D points	41k frames
RADDet [31]	2021	FMCW	ADC, RAD	2D boxes	10k frames
RadIal [32]	2021	FMCW	ADC	2D boxes	8k frames
Boreas [33]	2022	Pulse	RA*	3D boxes	7k frames

Table 3.1: Comparison between 2D radar object detection datasets. Datasets that only provide radar point cloud are not included. The size of the dataset does not include frames with no annotations. *2D tensors with no speed information.

The autonomous driving community saw a boom in datasets for object detection in the last few years, with the emergence of nuScenes [34], one of the first multimodal autonomous driving dataset. While nuScenes only provides radar point clouds, it has been a great source of inspiration for the radar community, leading to the creation of several datasets that provide radar data in different formats. Table 3.1 lists some of the most relevant datasets for 2D object detection using radar signals. RADIATE [30] and Boreas [33] are datasets collected using a scanning pulse radar, while the others use FMCW radars. As previously discussed in subsection 2.1.1, while scanning pulse radars provide a 360-degree view of the environment, they do not carry speed information, which is crucial for autonomous driving applications.

They are also less seen on recent production vehicles since their mounting position is limited to the top of the vehicle, which is not ideal for the design of modern vehicles. CARRADA [29], RADDet [31] and RadIal [32] are three datasets that provides flexibility in the choice of the detector's input format, with CARRADA providing RA, RD and RAD tensors, and the other two providing raw ADC signals which can be processed into other formats accordingly. However, the sizes of these datasets are relatively limited.

The CRUW dataset [4, 5] is one of the largest dataset for 2D object detection using FMCW radar signals, containing 400k synchronized frames of camera images and radar RA tensors sampled at 30Hz. The dataset is collected in various urban and suburban environments including highways, parking lots, and residential areas. Daytime and nighttime conditions are both covered, but adverse weather conditions such as rain and snow are not included. A small subset of the CRUW dataset containing 41k annotated frames (40 sequences) and 11k unannotated frames (10 sequences), is made public through the ROD2021 challenge (thus referred to as CRUW ROD2021 dataset). An illustrated overview of the dataset is shown in Figure 3.2. Two perpendicular 77 GHz FMCW radars are mounted on the top of the vehicle, calibrated and synchronized. The collected radar data is processed into RA tensors, with each tensor containing 128 range bins and 128 azimuth bins, covering a field of view of $\pm 90^\circ$ in azimuth and 1 to 25 meters in range. Each radar frame consists of 255 chirps, but only chirps 0, 64, 128, and 192 are provided in the dataset, resulting in a data shape of $4 \times 128 \times 128 \times 2$, where the last dimension represents the real and imaginary components. Annotations are provided in the form of 2D object center points with a class label of pedestrian, cyclist, or vehicle. The spatial distribution of the object samples from the annotated subset is visualized in Figure 3.3.

3.2.2 Evaluation Metrics

At a high level, 2D object detection from radar signals shows resemblances with 2D object detection from images, with both tasks aiming to localize and classify objects in a 2D plane.

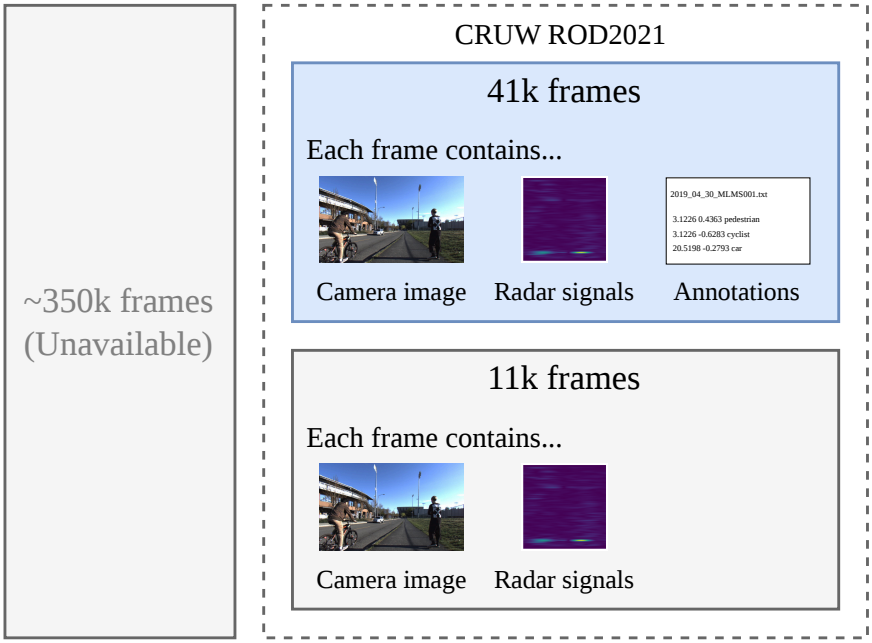


Figure 3.2: Overview of the CRUW dataset. The annotated subset of CRUW ROD2021 dataset is used in this thesis for training and evaluation. Camera images are only used for visualization purposes. Radar data is provided as RA tensors.

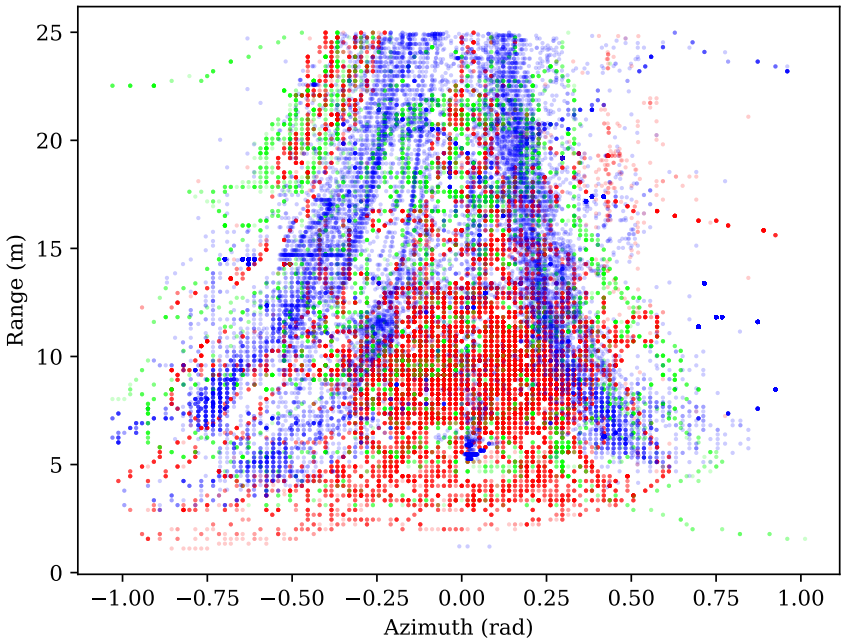
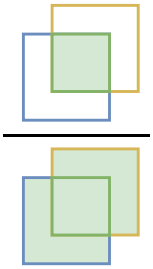


Figure 3.3: Data distribution of CRUW dataset. Each dot represents an object annotation in the dataset, ■ pedestrian, ■ cyclist or ■ car. The ego vehicle is located at the origin, facing upwards.

Thus, the evaluation of 2D radar object detectors shares some common metrics with image-based detectors, with the most prevalent metrics designed around the Intersection over Union (IoU) measurement. IoU measures how close the object prediction is to the ground truth, and takes into account the area of the ground truth and the prediction. Given the ground truth area of an object S_{gt} and the predicted area of the object S_p, the IoU is defined in Equation 3.1.

$$\text{IoU} = \frac{S_{gt} \cap S_p}{S_{gt} \cup S_p} = \quad (3.1)$$



The IoU of any two areas should be between 0 and 1, with 0 meaning no overlap and 1 meaning perfect overlap. The larger the IoU, the better the prediction is. IoUs are usually thresholded to determine whether a prediction is a match or a miss, higher threshold means a stricter evaluation, and vice versa.

With the predictions and ground truths associated through IoU thresholds, the bounding boxes (either predicted or ground truth) can be split into the following three exclusive sets:

- **True Positive (TP)**: correct detections, bounding boxes that present a match between a prediction and a ground truth;
- **False Positive (FP)**: incorrect detections, bounding boxes that are predicted but do not match any ground truth;
- **False Negative (FN)**: missed objects, ground truth bounding boxes that do not have any matching prediction.

Note that predictions with low confidence should be discarded, thus not included in TP or FP. Although the concept of True Negative (TN) exists in classification tasks, it is not applicable in the context of object detection since it is pointless to consider bounding boxes

that neither represent an object nor a prediction. It also will not be needed in the calculation of the following two metrics, *precision* and *recall*, as shown in Equation 3.2.

$$\begin{aligned} P &= \frac{|TP|}{|TP| + |FP|} = \frac{|TP|}{|DT|} \\ R &= \frac{|TP|}{|TP| + |FN|} = \frac{|TP|}{|GT|} \end{aligned} \tag{3.2}$$

where P and R are precision and recall, DT and GT are the set of detections and the set of ground truths, respectively. The two metrics are often used together to evaluate the performance of a detector, with high precision indicating that the detector is good at making correct predictions, and high recall indicating that the detector is good at finding all objects in the scene.

When looking into the calculation of precision and recall, one may notice that the two metrics depend heavily on the predetermined confidence threshold. As the confidence threshold decreases, the number of TP is bound to increase, leading to a higher recall, while the precision is likely to decrease as FP has a much higher potential to increase than TP. This negative correlation between precision and recall is often visualized in a PR curve, an example of which is shown in Figure 3.4.

The PR curve is a plot of precision against recall, with the confidence threshold as the parameter. Area Under Curve (AUC) is often used to summarize the performance of a detector, with a larger AUC indicating higher precision with higher recall, thus better performance. In practice, the curve is often interpolated to obtain a monotonic shape for less calculation complexity. This metric is referred to as Average Precision (AP) in literature, and is the most widely used metric for evaluating object detectors. It is worth noting that different datasets may have slightly different definitions for AP, e.g. the MS COCO dataset [35] defines AP as the average of AP at different IoU thresholds, from 0.5 to 0.95 with a step of 0.05. Since AP is calculated for each class separately, the overall performance of a detector is often summarized by mean Average Precision (mAP), the mean of AP across all

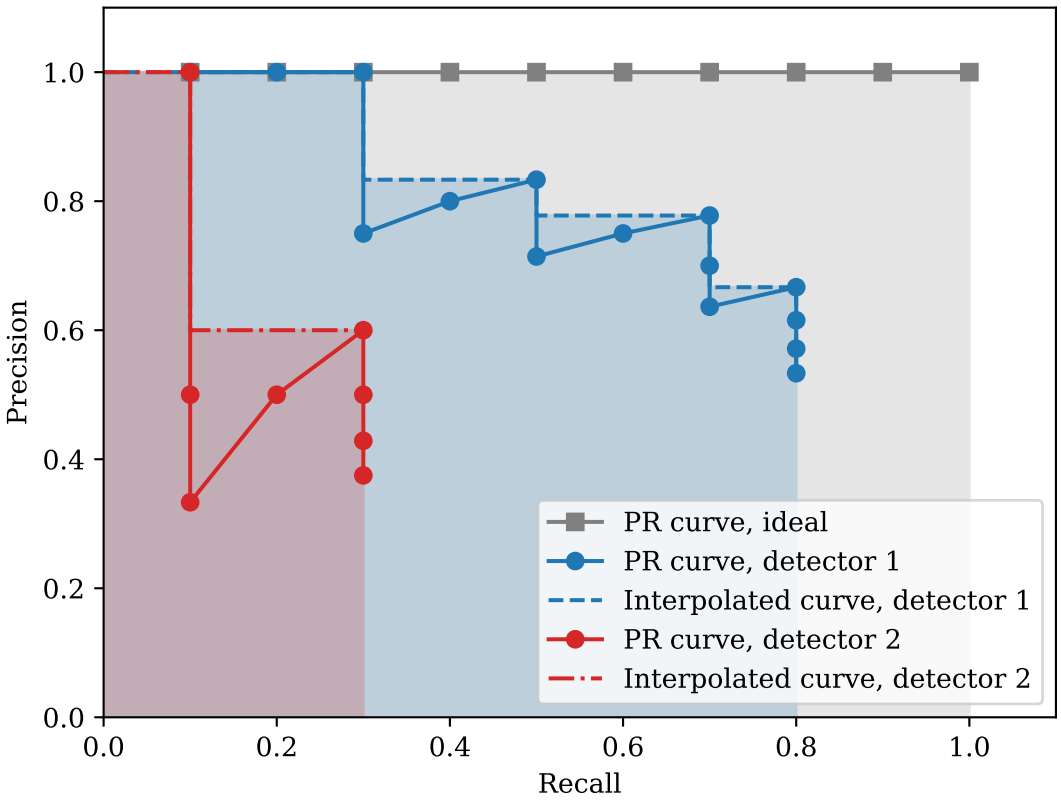


Figure 3.4: Example of PR curves. ■ An ideal detector would have a PR curve that passes the top right corner and an AUC of 1. ● Detector 1 shows a better performance than ● detector 2, as the AUC of its interpolated curve is larger than detector 2's.

classes. An Average Recall (AR) and mean Average Recall (mAR) metric is also defined in a similar manner.

For radar object keypoint detection problems, however, the IoU metric is not applicable since no bounding boxes are involved. In [4, 5], an Object Localization Score (OLS) metric is defined as an alternative to IoU, to describe the similarity between two detections, with considerations unique to the radar domain. The OLS metric is defined in Equation 3.3.

$$\text{OLS} = \exp\left(\frac{-d^2}{2(s\kappa_{cls})^2}\right) \tag{3.3}$$

where d is the distance between two points, s is the object distance from the radar, and κ_{cls} is a class-specific constant that represents the error tolerance for class cls, manually tuned to make OLS distributed between 0 and 1. OLS is more tolerant to distant objects than close objects, a property tailored for evaluating ADAS sensing systems, since objects closer to the ego vehicle are more relevant to the detector. In [4, 5], the AP is therefore calculated based on OLS instead of IoU, and averaged on different OLS thresholds from 0.5 to 0.9 with a step of 0.05, similar to the COCO AP metric.

3.2.3 Methods

Traditionally, radar object detection has been tackled using classical signal processing and machine learning techniques in a multi-stage pipeline. Due to the low resolution and noisy nature of radar signals, heuristic methods are often employed to first remove clutter and noise from the radar data, followed by a clustering algorithm, either model-based or learning-based, to group the positive responses into clusters representing different objects. An additional classification step is often needed to classify the detected objects into different categories. Constant False Alarm Rate (CFAR) [36] stands as the most widely used method for discriminating between noise and object responses in radar signals. For every Cell Under Test (CUT), CFAR estimates the noise level from a selection of reference cells in the neighborhood, and

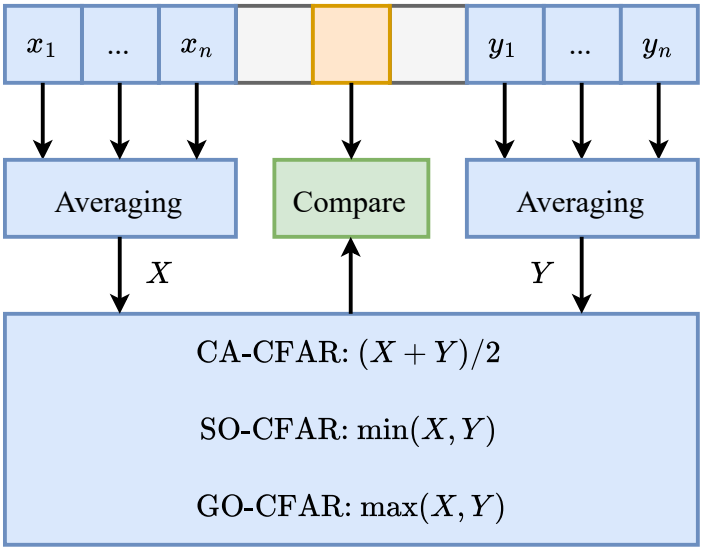


Figure 3.5: CFAR algorithm. The CUT is evaluated against a dynamic threshold calculated from the reference cells in the neighborhood. Guard cells are used to avoid spectral leakage from the CUT.

calculates a dynamic threshold based on the estimated noise level and a predefined false alarm rate. If the response in the CUT exceeds the threshold, it is classified as an object response; otherwise, it is classified as noise. The CFAR algorithm can be described as Figure 3.5.

CA-CFAR (Cell Averaging CFAR) is one of the simplest and most commonly used variants of CFAR, where the noise level is estimated by averaging the power of the reference cells. This method works well in homogeneous environments where the noise level is consistent across the reference cells. However, the false alarm rate may increase significantly in multiple target scenarios or non-homogeneous environments due to masking effects, leading to missed detections. SO-CFAR (Smallest Of CFAR) and GO-CFAR (Greatest Of CFAR) act as two simple solutions to this problem by splitting the reference cells into two halves, and using the smaller or greater noise estimate from the two halves, respectively. Each of these methods has its own advantages and disadvantages, and the choice of method often depends on the specific application and environment. More advanced CFAR techniques have also been proposed to address the limitations of these basic methods, such as OS-CFAR (Or-

dered Statistic CFAR), which uses order statistics to estimate the noise level, making it more robust to outliers and multiple target scenarios. In 2D radar spectrums, a naive extension is to apply 1D CFAR along each dimension sequentially, while more advanced methods such as CA-CFAR with 2D sliding windows have also been proposed to better capture the spatial correlations in the data [37].

With the point cloud generated from CFAR, clustering algorithms need to be applied to group the points into object entities. Density Based Spatial Clustering of Applications with Noise (DBSCAN) [38] is one of the most widely used clustering algorithms in radar object detection tasks [39, 40]. By recursively grouping points that are closely packed together, DBSCAN is able to identify clusters of arbitrary shapes and sizes, while effectively filtering out outliers as noise. However, DBSCAN requires careful tuning of its hyperparameters, such as the neighborhood radius and the minimum number of points required to form a cluster, which can significantly affect its performance.

With the advancement of deep learning techniques, end-to-end radar object detection methods based on CNNs and Transformers have gained popularity in the past decade. In subsection 2.1.2, an overview of the commonly used radar data representations was provided, including raw ADC signals, as well as RA, RD, and RAD tensors. These representations are often interchangeable since they essentially carry the same information, only in different domains. They can also be treated equally in the sense of dimensionality. Since objects are characterized by high responses in the temporal domain, a common practice is to treat these representations as 2D images, and existing image-based object detection architectures can be applied. This has been proven to be fruitful in literature. RADDet [31] uses a ResNet backbone to extract features from RAD tensors, and two YOLO heads to generate 3D and 2D bounding box predictions on the polar and Cartesian domains, respectively. The backbone is trained with the 3D RAD YOLO head, after which the 2D YOLO head is trained with the backbone frozen. RODNet [4, 5] further extends the idea by using a hourglass-shaped 3D CNN to model the temporal relations between frames. A sequence of

RA maps are fed into the network, and the output is sets of confidence maps for each frame, each representing an object class. Temporal modelling through recurrent networks has also been explored in [41, 42], where Long-Short Term Memory networks are appended to CNN-based feature extractors to achieve late fusion of temporal information. While these methods have shown promising results, they are limited by the inherent constraints of CNNs, such as local receptive fields and lack of global context modelling.

With the emergence of vision Transformers, the radar community has also started to explore the use of Transformers for object detection, especially Swin Transformer [2]. Swin Trasformer offers a hierarchical architecture capable of extracting features at different scales, its windowed attention mechanism reduces the heavy complexity of the dense attention mechanism seen in ViT, and allows information exchange between adjacent windows through window shifting. In T-RODNet [43], one of the first attempts to use Swin Transformer for 2D object detection from radar signals, the author constructs a hybrid U-Net-like architecture with Swin Transformer blocks, and two strategies are employed for information exchange between the encoder and decoder: skip connections and cross attention. On the other hand, T-FFTRadNet [44] is based on an encoder-only architecture seen in the original Swin Transformer, where the features extracted from the encoder are upscaled to the original resolution and fed into a MLP head for detection. Notably, T-FFTRadNet explores the use of raw ADC signals as input, and uses a complex-valued MLP to approximate FFT operations required to generate the RAD cube, thus avoiding the computation overhead of the 3D-FFT.

Despite its success in various fields, Transformers are not without drawbacks, particularly due to the attention mechanism's $O(N^2)$ complexity. Different architectures have since been proposed to address these problems from different perspectives, and efforts to apply these new architectures to radar object detection have also been made. Mamba [45], for example, is a novel architecture that strikes a balance between the performance of $O(N^2)$ models (e.g. Transformers) and the efficiency of $O(N)$ models (e.g. recurrent networks). By

storing the context information selectively, Mamba achieves sub-quadratic complexity without sacrificing long range dependencies. Its variant in the computer vision domain, Vision Mamba, has shown promising results on radar object detection tasks [46], and early explorations of Mamba for radar object detection have also been made [47, 48, 49]. MetaFormer [3], on the other hand, tries to solve the problem from a so-called "physics-style" approach, as discussed in subsection 2.2.2. By replacing the attention mechanism with other token mixers, MetaFormer achieves similar performance to Transformers with more efficiency, and has been shown to be effective in radar object detection tasks [50, 51]. Literature has also explored the use of hybrid architectures that combine the strengths of multiple architectures. For instance, CNNs have been used as local feature extractors, while Transformers or MetaFormers are used for global context modelling [52, 53]. These hybrid architectures have shown promising results in radar object detection tasks, leveraging the advantages of both architectures to achieve better performance.

Chapter 4

Spatio-Temporal Radar Object Detection

This chapter describes the design and implementation of a 3D radar object detection model based on Swin Transformer [2]. Its overall architecture and its temporal modeling capabilities are discussed, along with the detection head and output space design. Experiments are conducted to evaluate the model's performance, with a focus on explainability.

4.1 Overall Architecture

An overview of the proposed architecture is presented in Figure 4.1. The overall shape of the architecture resembles U-Net [54], a CNN architecture widely adopted in various computer vision tasks. Although U-Net was originally designed to produce segmentation masks, our

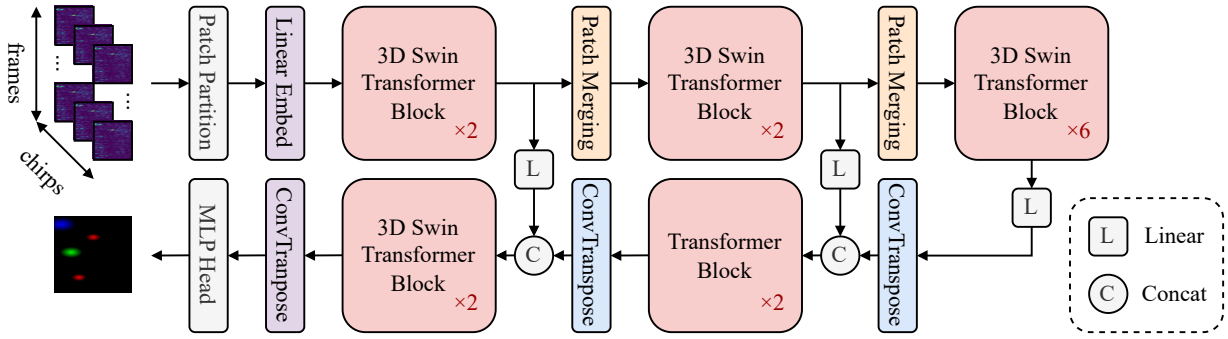


Figure 4.1: Overview of the backbone architecture.

heatmap-based keypoint detection strategy shares similarities with the segmentation task, which will be further discussed in section 4.3. With the prevailing success of vision transformers in various vision tasks, it has become increasingly common in the medical image segmentation field to improve the original U-Net architecture by replacing the CNN layers with transformer blocks [55, 56, 57]. Our model takes a similar approach by utilizing Swin Transformer blocks as the main building units for both the encoder and the decoder.

The model takes a sequence of RA tensors as input, with tensors from different chirps stacked along the channel dimension. Performing object detection directly on the RA dimensions is the most intuitive way, since the goal is to acquire the 2D location of objects in polar coordinates, or Cartesian coordinates after conversion. Otherwise, the model would need to learn a transposition to align the dimensions, which is not desirable when designing neural networks. The RA tensors also resemble a Bird's Eye View (BEV) representation of the environment, which has been shown to be effective for various perception tasks [58, 59, 60], with the only difference being the use of polar coordinates instead of Cartesian coordinates. It is also desirable to prioritize objects that are closer to the ego vehicle, since they are more likely to interact with the ego vehicle, and their future trajectories are more likely to intersect with the ego vehicle's trajectory. This is naturally achieved on the polar plane due to higher point density in the near field. Note that apart from the polar RA dimensions and the channel dimension, a third temporal dimension is introduced to facilitate noise cancellation and temporal modeling, which will be discussed in section 4.2. Since radar spectrums are complex-valued, they can't be directly processed by real-valued neural networks. Efforts have been made to preprocess complex-valued radar data using complex-valued neural networks, but results show that they fail to achieve comparable performance to using converted real-valued inputs [44]. Therefore, we opt to simply separate the real and imaginary components of the RA tensors along the channel dimension, resulting in a real-valued input tensor with $2C$ channels, where C is the number of chirps.

Inspired by vision transformers, a token splitting module and a linear embedding module

are used to partition the RA tensors into smaller patches, and then project them into linear tokens. Their structures are identical to those in the original Swin Transformer, with the exception that the number of channels had to be changed. The encoder and the decoder are divided into stages, where each stage incorporates several Swin Transformer blocks and a transition layer. The Swin Transformer blocks are organized in an alternating manner to allow information exchange across windows, which will be further discussed in section 4.2. In each stage, adjacent tokens are merged together in the transition layer to reduce the spatial resolution by half, while the number of channels is doubled to maintain the model capacity. Note that tokens from adjacent frames are not merged together, as the temporal resolution is preserved throughout the entire backbone to facilitate temporal modeling. The decoder mirrors the encoder, but with the token length halved at each stage, and transposed convolution layers are used for upsampling. Stages at the same "depths" are connected through skip connections, a characteristic design from U-Net. With skip connections, tokens with the same spatial resolution and same level of abstraction are fused in the decoder. This design helps recover fine-grained details that may be lost during downsampling in the encoder, which is crucial for accurate object localization. At the end of the decoder, the spatial resolution of the tokens is restored, and a final MLP head is used to project the feature maps into the output space.

4.2 Temporal Modeling with 3D Windowed Attention

Swin Transformers are characterized by its **S**hifted **win**dow mechanism, which achieves a balance between computational efficiency and global context modeling. Figure 4.2 (a) illustrates the structure of two consecutive Swin Transformer blocks, their key difference from the original Transformer block is the use of window-based attention, which is illustrated in Figure 4.2 (c).

The first block is a regular Window Multi-Head Self-Attention (W-MSA) block, in which

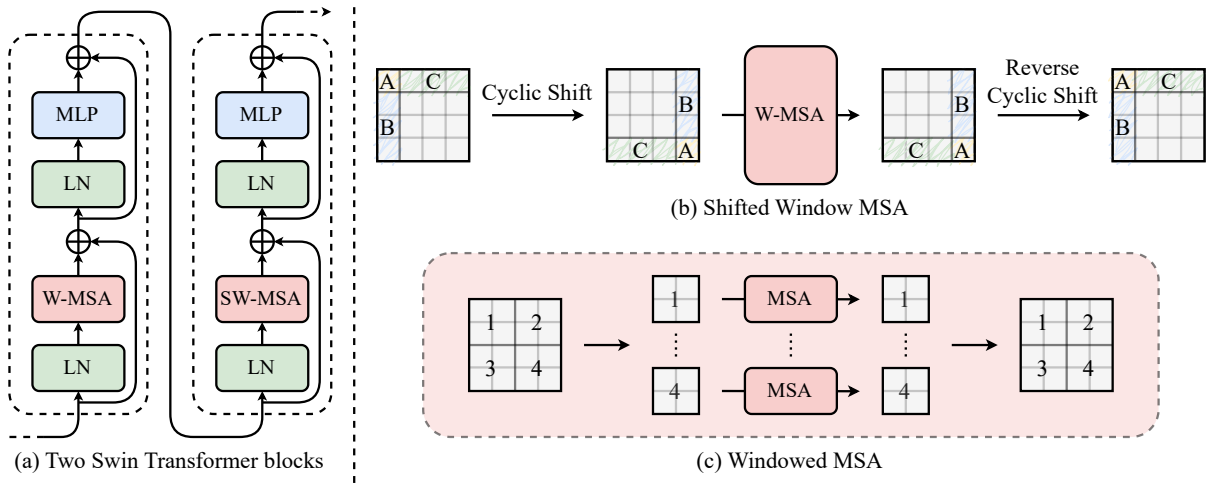


Figure 4.2: Swin Transformer block and its window-based attention. (a) Two consecutive Swin Transformer blocks, their structures are identical except for the shifted windows. (b) In SW-MSA, the tokens are shifted along each dimension for W-MSA, and the shift is reversed afterwards (c) In W-MSA, MHSA is applied to each window independently, resulting in local reception fields with lower computational cost.

the input tokens are partitioned into non-overlapping windows based on their spatial locations, and Multi-Head Self-Attention (MHSA) is applied to each window independently. This strategy is based on the prior knowledge that local features in images are often more relevant than global ones, the same assumption that powers the success of CNN over MLP. Compared with the global attention mechanism found in the original transformers, the computation complexity of this operation is therefore reduced from $O(N^2)$ to $O(N^2/W)$, where N is the number of tokens and W is the number of windows. While this strategy significantly improves computational efficiency, it also limits information exchange between windows, and objects located on the edge of windows might not be fully captured. To mitigate this issue, the second block introduces Shifted Window Multi-Head Self-Attention (SW-MSA), where the windows are cyclic-shifted by half of their size in each dimension before applying W-MSA, as illustrated in Figure 4.2 (b). This shifting operation allows tokens previously located in different windows to be grouped together, and enables cross-window information propagation, as well as better modeling of objects on window edges. After the attention operation, the shift is reversed to restore the original token arrangement. By alternating be-

tween W-MSA and SW-MSA, the model can effectively capture both local and cross-window features, while still maintaining computational efficiency. Each Swin Transformer block can therefore be described as Algorithm 1.

Algorithm 1 Swin Transformer Block

function SwinTransformerBlock(X, m, n, $shifted$) ▷ *Input tokens: X, number of windows: m, number of heads: n, whether to shift windows: $shifted$*
  $X' \leftarrow \text{LayerNorm}(X)$
  if $shifted$ **then**
    $X' \leftarrow \text{CyclicShift}(X')$ ▷ *Cyclic shift tokens by half window size*
  end if
  $\{X_w | w < m\} \leftarrow \text{PartitionWindows}(X')$ ▷ *Partition and reshape tokens*
  for all X_w **do**
    $Q_w, K_w, V_w \leftarrow L_{qkv}(X_w)$ ▷ *Linear projections*
    $\{Q_{w,h}, K_{w,h}, V_{w,h} | h < n\} \leftarrow \text{SplitHeads}(Q_w, K_w, V_w)$
    for all $Q_{w,h}, K_{w,h}, V_{w,h}$ **do**
      $A_{w,h} \leftarrow \text{Softmax}(Q_{w,h} K_{w,h}^T / \sqrt{d} + B)$ ▷ *B is relative position bias*
      $Y_{w,h} \leftarrow A_{w,h} V_{w,h}$
    end for
    $Y_w \leftarrow \text{ConcatHeads}(\{Y_{w,h} | h < n\})$
    $Y_w \leftarrow L_{out}(Y_w)$ ▷ *Linear projection*
  end for
  $Y \leftarrow \text{MergeWindows}(\{Y_w | w < W\})$ ▷ *Merge and reshape tokens*
  if $shifted$ **then**
    $Y \leftarrow \text{ReverseCyclicShift}(Y)$ ▷ *Restore original arrangement*
  end if
  $X'' \leftarrow X + Y$ ▷ *Residual connection*
  $X''' \leftarrow \text{LayerNorm}(X'')$
  $Y \leftarrow \text{MLP}(X''')$
  $Z \leftarrow X'' + Y$ ▷ *Residual connection*
  return Z
end function

In ADAS applications, consistent detection performance over time is crucial for downstream tasks such as object tracking and trajectory prediction, and it is therefore important to incorporate temporal modeling capabilities into radar object detection, transforming the task from a "frame to frame" problem to a "sequence to sequence" problem. Some works have taken a recurrent approach to temporal modeling, where features from previous frames are fused into the current frame using temporal self attention [60] or simple concatenation

[61]. However, these methods heavily rely on the ego vehicle's motion projection to align the features across time, which is calculated from inertial measurements and not always accurate. Additionally, it is often challenging to fuse features recurrently from a larger temporal window. Considering these limitations, we propose to solve the temporal modeling problem through a native extension to the attention mechanism. By stacking RA tensors from multiple frames along a new temporal dimension, we can extend the 2D window-based attention to 3D window-based attention, where the windows are now 3D cubes spanning the spatial and temporal dimensions. This temporal modeling strategy creates no conflict with the prior knowledge Swin Transformer is based on — local features are more relevant — since object motions are continuous, and tokens that are close in time are more relevant than those that are far apart. Additionally, this strategy allows for direct modeling of longer temporal windows without the need for ego motion compensation, as the attention mechanism can learn to focus on relevant tokens across time. As a bonus, this approach also allows for automatic noise suppression, as temporal attention can selectively attend to consistent signals and ignore transient noise. This extended 3D attention mechanism is applied throughout the entire backbone, enabling comprehensive spatio-temporal feature extraction, and the model can efficiently generate detection outputs for all frames in the input sequence with a single forward pass.

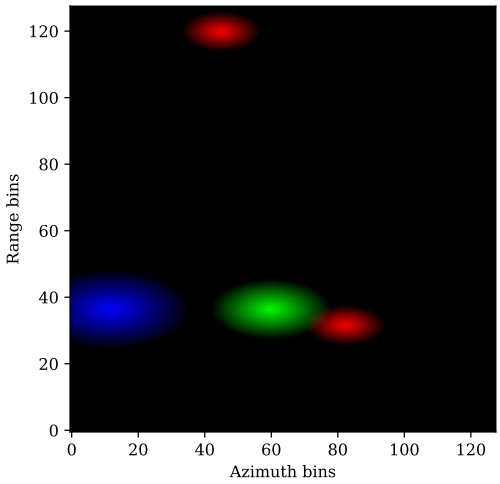
4.3 Detection Head and Output Space

In this thesis, we focus on an object keypoint detection problem for low-definition radars, where the goal is to predict the 2D locations of object centers, instead of bounding boxes. Inspired by the success of heatmap-based keypoint detection methods [62], we employ a similar strategy for our detection head and output space design. We construct a confidence map with multiple channels in a polar coordinate that corresponds to the input RA map. Each object in the scene is represented by a Gaussian kernel, whose variance is proportional

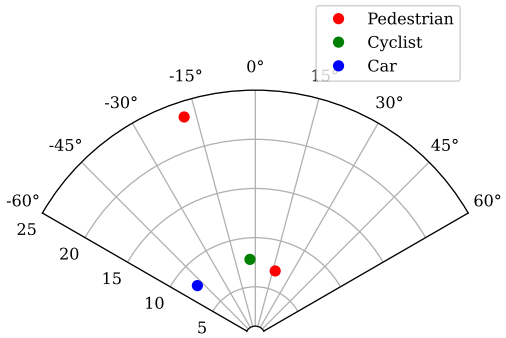
to the object size. The kernels are placed with their ground truth polar coordinates on the corresponding class channel, creating a confidence map that highlights the object's position. An example of the confidence map is shown in Figure 4.3. Additionally, the kernel size is adjusted based on the object's distance to reflect the varying point density in different range bins, in line with [4, 5]. A simple MLP head is therefore used to project the feature maps from the decoder into this output space to approximate the confidence map. Finally, object locations are extracted through peak detection and NMS, which will be detailed in section 4.4.

Compared to query-based detection methods [26, 27, 28, 60], this strategy offers a dense supervision signal as each pixel in the output space contributes to the loss calculation, hence producing a faster and more stable regression. This strategy also avoids the need for bipartite matching during training, which can lead to training instability [27].

Notably, T-FFTRadNet [44] employs a similar output space design based on confidence maps, but in a YOLO [22] style. A detection head is first used to approximate a coarse occupancy map, where each object is represented by a square kernel with a fixed size. To compensate for the precision loss from quantization, a regression head is then used to predict correction factors in both range and azimuth dimensions, to achieve a higher resolution. At last, a classification head is used to produce a likelihood map for each object class, and the class of an object at a given location is determined by the channel with the highest value. This YOLO style design requires a manually tuned multi-objective loss, which is not only more complex but also less stable than the single-head design used in this work. Additionally, the square kernel design is not optimal for regressing object locations, and may introduce additional bias in the peak detection process.



(a) Visualization of confidence map



(b) BEV of the scene in polar coordinates

(c) Front view of the scene

Figure 4.3: Example of confidence map for supervision, generated from the 96th frame of sequence `2019_04_30_MLMS001`. It is constructed on a polar coordinate aligned with the radar's RA dimensions. Objects from different classes, e.g. ■ pedestrian, ■ cyclist and ■ car, are placed on different channels of the confidence map.

4.4 Implementation Details

The CRUW ROD2021 dataset [4, 5] is selected for training and evaluation of the proposed model, its specifications are detailed in subsection 3.2.1. Since only 40 out of 50 sequences are annotated, we split the annotated subset into training and validation sets with a ratio of 9:1, e.g. 36 sequences for training and 4 sequences for validation, in line with previous works [43, 50], to prevent data leakage and ensure a fair comparison.

To enhance the model's generalization capabilities on small datasets, data augmentation techniques can be applied during training. Common augmentation methods for computer vision tasks include random flipping, rotation, scaling, and color jittering. Radar RA spectra, however, are represented in polar coordinates instead of cartesian coordinates, and they do not necessarily share the same prior assumptions that exist in natural images. Thus it is crucial to avoid augmentations that violate radar signal distribution and properties. In this work, we apply random flipping along the azimuth dimension with a probability of 0.5, as objects that appear on the left side of the ego vehicle should carry the same signatures as those on the right side. Additionally, random flipping along the temporal dimension is also applied with a probability of 0.5. It is worth noting that when temporal flipping is applied, the order of the chirps within each frame must also be reversed to maintain the correct mapping between chirp indices and time. We opt not to apply jittering or noise injections, as these may alter the value distribution of the radar signals and introduce unwanted bias.

For each radar frame, a prenormalized RA tensor of shape shape $4 \times 128 \times 128 \times 2$ is loaded from the dataset, and rearranged into a 3D tensor of shape $128 \times 128 \times 8$ by stacking the real and imaginary components of the 4 chirps along the channel dimension. A sliding window of 16 frames is used, and frames are stacked along the temporal dimension to form a 4D tensor of shape $16 \times 128 \times 128 \times 8$.

We built our codebase with the PyTorch 2 [63] deep learning framework on Python 3.13. The model is trained and evaluated on a single Nvidia GeForce RTX 4090 GPU with 24 GB of memory. Loss is calculated with the Mean Squared Error (MSE) loss described in

Equation 4.1.

$$\mathcal{L}_{MSE} = \frac{1}{N}\sum_{i=1}^{N}\left(\hat{Y}_i - Y_i\right)^2 \tag{4.1}$$

where $\hat{Y}$ is the predicted confidence map, Y is the ground truth confidence map, and N is the total number of elements in the confidence map. The AdamW optimizer [64] is used to update the model with an initial learning rate of 1×10^{-4}, which decays to 0 with a cosine schedule. A batch size of 2 is used, and the model is trained for 20 epochs.

During the evaluation process, 8-neighbor peaks are extracted from the output confidence map as detection candidates, as shown in Figure 4.4. The candidates are then filtered by a Location-based Non-Maximum Suppression (L-NMS) algorithm [4, 5], which is described in Algorithm 2. For each object class, the L-NMS algorithm iteratively selects the most confident candidate and removes other candidates that have an OLS value larger than a predefined threshold with respect to the selected candidate, until all candidates are processed. It acts as a replacement to common IoU-based NMS for point-based radar object detection tasks by incorporating OLS, an object similarity metric tailored for the radar field described in subsection 3.2.2.

Algorithm 2 Location-based Non-Maximum Suppression (L-NMS)

function L-NMS($\mathcal{P}$) ▷ *Input candidate peak set* $\mathcal{P}$
  $\mathcal{P}^* \leftarrow \emptyset$ ▷ *Initialize final peak set* $\mathcal{P}^*$
  while $\mathcal{P} \neq \emptyset$ **do**
    $p \leftarrow \max(\mathcal{P})$ ▷ *Find the most confident peak*
    $\mathcal{P}^* \leftarrow \mathcal{P}^* \cup \{p\}$
    $\mathcal{P} \leftarrow \mathcal{P} - \{p\}$ ▷ *Move p from candidate set to final set*
    for all $p' \in \mathcal{P}$ **do**
      if $\mathrm{OLS}(p, p') > threshold$ **then**
        $\mathcal{P} \leftarrow \mathcal{P} - \{p'\}$ ▷ *Remove* p' *from candidates*
      end if
    end for
  end while
  return $\mathcal{P}^*$ ▷ *Return final peak set* $\mathcal{P}^*$
end function

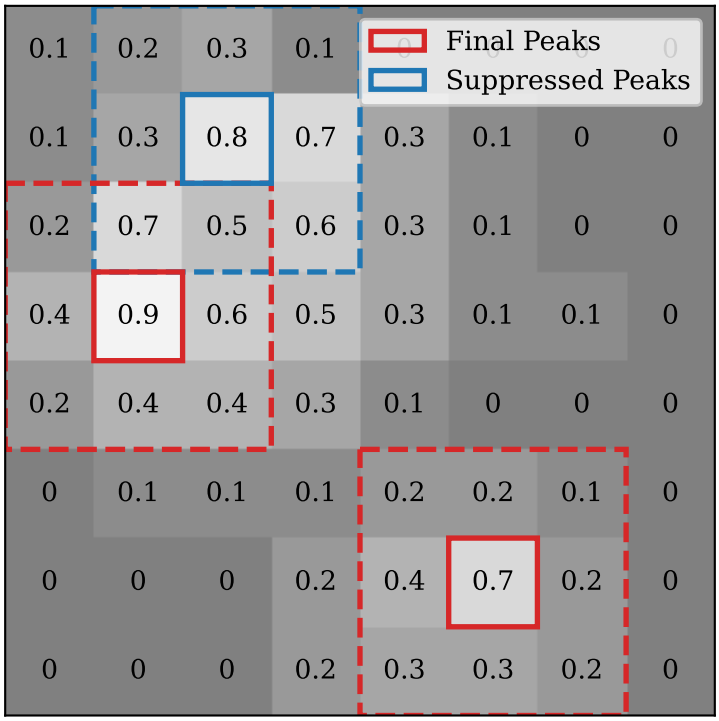


Figure 4.4: Example of L-NMS results. The most confidence peak is first selected, and OLS is computed between it and the other two candidates. The second candidate is removed since its OLS with the first peak is larger than threshold, while the third candidate is kept.

Table 4.1: Class-wise evaluation at different OLS thresholds.

OLS Thresh.	AP	AR	AP	AR	AP	AR	mAP	mAR
0.5	89.68%	94.20%	92.39%	95.64%	94.67%	98.63%	91.82%	95.81%
0.6	89.05%	93.76%	91.36%	94.54%	92.65%	97.12%	90.70%	94.89%
0.7	87.72%	92.83%	90.09%	93.39%	89.09%	95.18%	88.79%	93.63%
0.8	83.17%	89.15%	86.83%	90.82%	83.32%	91.30%	84.30%	90.23%
0.9	62.40%	75.87%	71.67%	79.72%	61.05%	75.92%	64.81%	77.04%
Average	83.57%	89.85%	87.38%	91.38%	85.34%	92.51%	85.18%	91.02%

4.5 Quantitative Evaluation

The proposed model is evaluated on 4 test sequences: `2019_04_09_BMS1001`, `2019_04_30_MLMS001`, `2019_05_23_PM1S013` and `2019_09_29_ONRD005`, which are excluded from the training set. The evaluation is conducted using AP and AR metrics based on OLS, as described in subsection 3.2.2. The model's performance in different object classes is reported in Table 4.1, with different OLS thresholds, ranging from 0.5 to 0.9 with a step size of 0.1. The average performance across all thresholds is also reported to provide a comprehensive overview of the model's capabilities.

From Table 4.1, the proposed model demonstrates similar performance across different object classes, with slightly lower performance on pedestrians compared to cyclists and cars. This is expected, as pedestrians are generally harder to detect due to their smaller size and lower radar cross section. While the model achieves the highest AR on cars, the AP is slightly lower than that of cyclists. This indicates that while the model is able to detect most cars, its accuracy in localizing them could be improved, possibly because of their larger size and faster speed.

To further analyze the model's performance in different driving scenarios, we break down the results by test sequences, as shown in Table 4.2. Sequences `2019_04_09_BMS1001` and `2019_05_23_PM1S013` are recorded in an empty parking lot with pedestrians and cyclists walking around, while `2019_04_30_MLMS001` is recorded on the sidewalk of a campus road with moderate traffic, and `2019_09_29_ONRD005` is recorded on a highway with fast-moving

Table 4.2: Class-wise evaluation in different sequences.

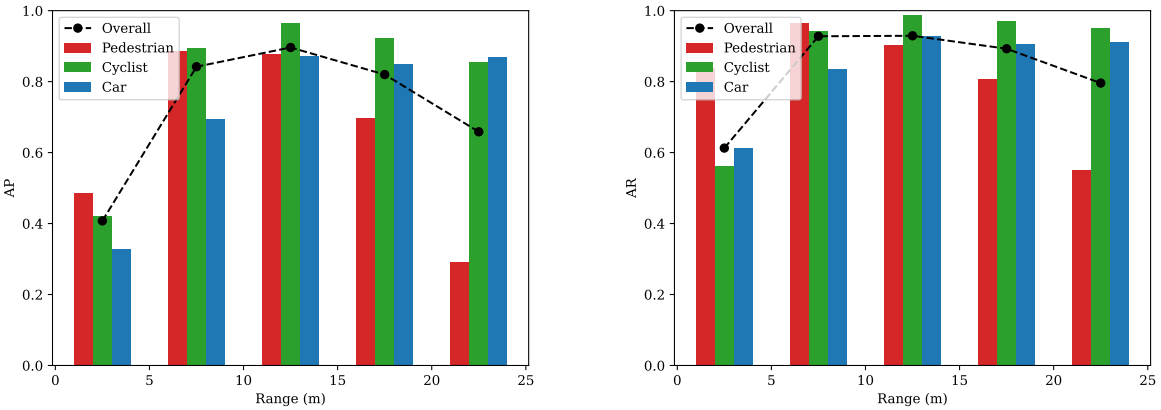
Sequence	Scene	Class	AP	AR
2019_04_09_BMS1001	Parking lot	Cyclist	88.94%	91.01%
		Mean	88.94%	91.01%
2019_04_30_MLMS001	Internal road	Pedestrian	77.71%	84.02%
		Cyclist	82.28%	90.86%
		Car	88.14%	94.07%
		Mean	80.86%	87.64%
2019_05_23_PM1S013	Parking lot	Pedestrian	97.59%	98.79%
		Cyclist	98.78%	99.31%
		Mean	97.69%	98.83%
2019_09_29_ONRD005	Highway	Car	84.44%	91.70%
		Mean	84.44%	91.70%

vehicles.

From Table 4.2, the model achieves the best performance in the parking lot scenarios, likely due to a relatively static environment with fewer occlusions and less clutter. The highway scenario yields median performance, as the fast-moving vehicles present challenges in terms of motion blur and rapid changes in radar signatures. The internal road scenario results in the lowest performance, especially for pedestrian and cyclist, an example of the scene is shown in Figure 4.3 (c). Due to the absence of textural information in radar signals, distractions from surrounding infrastructure, such as trees, signposts, and other static objects, may carry similar radar signature as the objects in interest, and introduce noise and false positives. This makes it harder for the model to accurately detect and localize pedestrians and cyclists.

4.5.1 Range Perception

Perception of objects at distance stands as a critical challenge for radar-based object detection, due to larger signal attenuation and smaller object signature at longer range. To evaluate the model's performance at different ranges, we divide the test set into several range



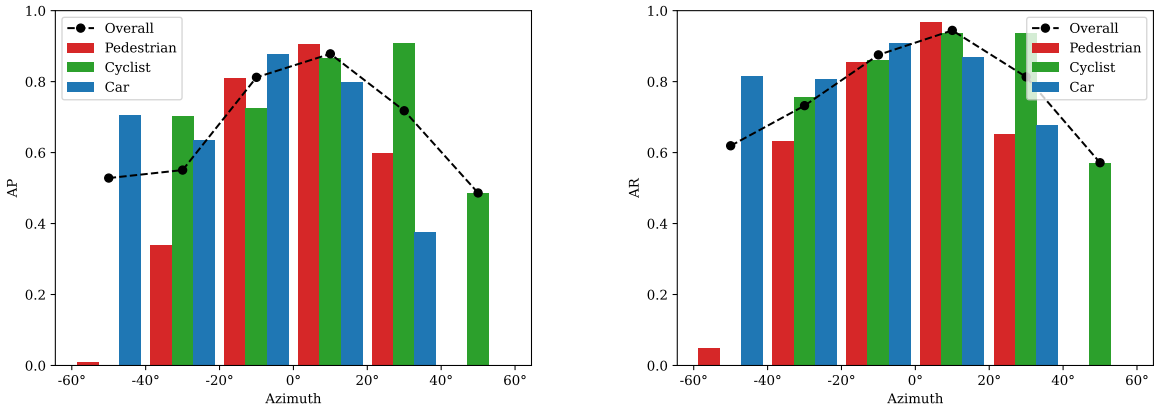
(a) AP at different range intervals

(b) AR at different range intervals

Figure 4.5: Perception performance at different ranges. AP and AR for each class are calculated for objects within different range intervals, as well as the overall mAP and mAR.

intervals, and calculate AP and AR for each class within these intervals, as well as the overall mAP and mAR. The results are presented in Figure 4.5.

From Figure 4.5, the model achieves the best performance at a medium range of 10 to 15 meters. Performance drops off at both close and long ranges. The decrease in performance at close range (0–5 meters) may be attributed to the limited number of training samples in this range, as observed in Figure 3.3. Interestingly, pedestrian detection at close range demonstrates a higher AR, possibly due to more pedestrian samples contained in the dataset at close range compared to other classes. It is worth noting that, while pedestrian AP is lower at close range, this does not necessarily imply poor localization accuracy, since the object distance acts as a scaling factor in the OLS metric, which means stricter localization requirements for closer objects. As for long range perception (beyond 15 meters), we observe a trend of proportional decline in detection performance to object size. Pedestrians, being the smallest objects, experience the most significant performance drop, followed by cyclists, while cars maintain relatively stable performance even at longer distances. This indicates that smaller radar cross sections at longer ranges stands as the primary challenge for long-range radar perception, and smaller objects are more adversely affected.



(a) AP at different azimuth intervals

(b) AR at different azimuth intervals

Figure 4.6: Perception performance at different azimuths. AP and AR for each class are calculated for objects within different azimuth intervals, as well as the overall mAP and mAR.

4.5.2 Angular Perception

The angular resolution of a multi-input FMCW radar system is determined by the number of receiving antennas and their physical arrangement. The CRUW dataset uses a radar system with 4 receiving antennas, providing a limited angular resolution, and poses challenges for accurate angular perception. To evaluate the model's performance at different azimuth angles, we divide the test set into several azimuth intervals, and calculate AP and AR for each class within these intervals, as well as the overall mAP and mAR. The results are presented in Figure 4.6. Due to limited samples at extreme angles, azimuth intervals with no samples in a particular class appear as gaps in the corresponding plots.

From Figure 4.6, the model shows a distribution of performance that peaks around the center azimuth (0 degrees) and gradually decreases towards the sides. This trend is consistent across all object classes, with pedestrians experiencing the most significant performance drop at extreme angles, followed by cyclists, while cars maintain relatively stable performance. FMCW radars typically have a narrower beamwidth in the center, providing better angular resolution and signal strength for objects located directly in front of the radar. As objects move towards the sides, the radar's beamwidth widens, leading to reduced angular resolution

and weaker signal returns. This results in decreased detection performance, particularly for smaller objects like pedestrians and cyclists. Theoretically, the model's performance should be symmetric about the center azimuth since radar signals should be flip-invariant, and horizontal random flipping during training should further enforce this property. The slight asymmetry observed in Figure 4.6 may be attributed to the non-uniform distribution of samples across different azimuth angles in the testing sequences.

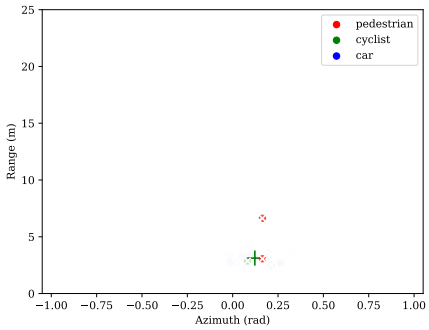
4.6 Qualitative Research

In this section, the proposed model is evaluated through non-numerical means, including case studies of detection results in representative scenarios, and visualization of attention maps from the Swin Transformer backbone for interpretability. These qualitative analyses aim to develop a deeper understanding of the model's decision-making process, identify potential failure modes, and provide entry points for improvements.

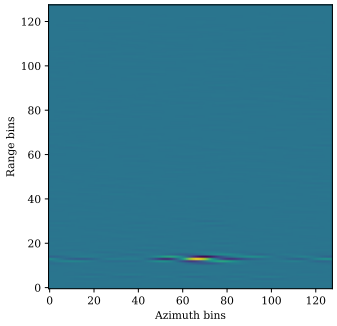
4.6.1 Results Case Studies

To gain insights into the model's strengths and weaknesses in different scenarios, we conduct case studies of detection results on selected representative frames from the test sequences. The camera image is provided alongside the radar input and output maps for better interpretability. Ground truth annotations, object prediction candidates, and final detections after NMS are plotted in polar coordinates, with ground truths represented as ✚ plus signs, prediction candidates as ● solid circles, and final detections as ✖ solid circles with white cross marks. Four representative cases are presented in Figure 4.7, Figure 4.8, Figure 4.9 and Figure 4.10.

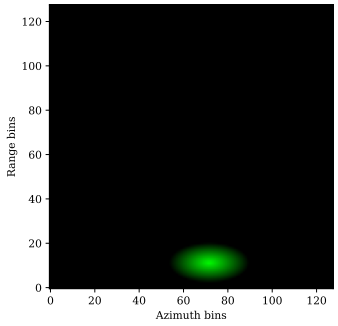
(a) Camera image



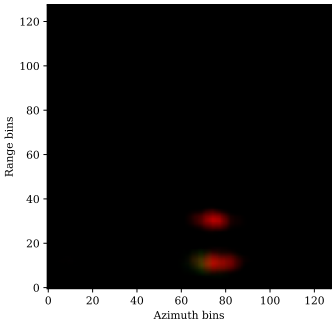
(b) Detection results



(c) Input radar spectrum



(d) Confmap ground truth

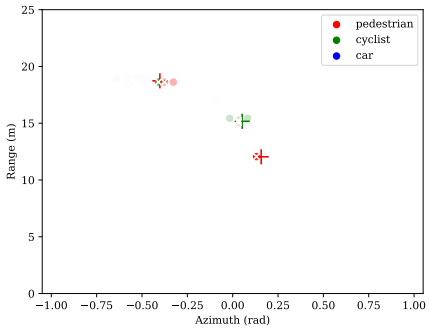


(e) Confmap prediction

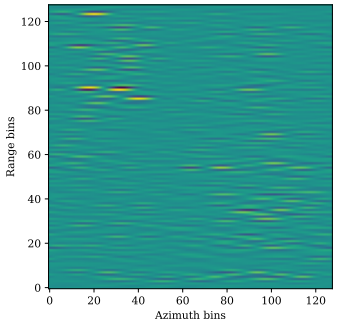
Figure 4.7: Case study of Sequence `2019_04_09_BMS1001, frame 72`. This frame is captured while a cyclist circles around in an empty parking lot. The cyclist is falsely detected as a pedestrian, and generates two duplicate detections.

In Figure 4.7, a relatively simple scenario is presented, where a single object in the near range is placed against an uncluttered background. However, the model misclassifies the cyclist as a pedestrian and generates an extra detection at a farther range. From the output confidence map shown in Figure 4.7 (e), it can be observed that the model is uncertain about the object's class, as the detection appears blurry in both object channels. The misclassification might be attributed to the similarity in radar signatures between pedestrians and cyclists, especially when the cyclist is moving slowly. The reason behind the duplicate detection, on the other hand, might be related to the reflection characteristics of the radar signal. The reflected wave from the cyclist could have echoed off the ego car, creating a secondary signal that the model mistakenly interprets as another object, as evident from the weaker response in the radar spectrum shown in Figure 4.7 (c).

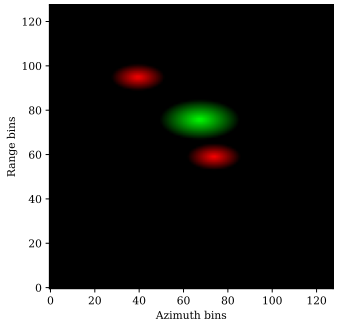
(a) Camera image



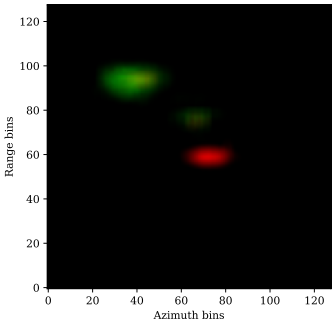
(b) Detection results



(c) Input radar spectrum



(d) Confmap ground truth

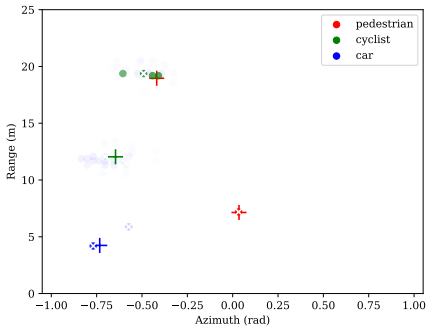


(e) Confmap prediction

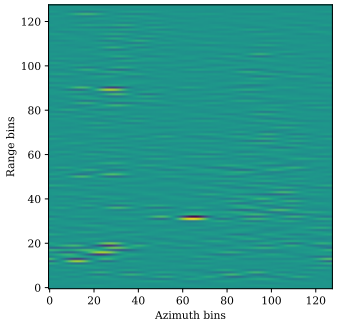
Figure 4.8: Case study of Sequence `2019_04_30_MLMS001, frame 224`. This frame is captured on a campus road with pedestrians and cyclists passing by. The pedestrian across the road is misclassified as a cyclist, and the model shows low confidence in detecting the cyclist.

In Figure 4.8, a more complex scenario is presented, where objects of different classes scatter at different ranges against a noisy background. Once again, the model shows uncertainty in classifying pedestrians and cyclists, but for a different reason. The pedestrian across the road is misclassified as a cyclist, likely due to noise and interference from bushes behind, which might generate a larger radar cross section enough to mimic that of a cyclist. On the right side, the cyclist moving away from the radar is detected with low confidence, with weak response in the output confidence map on both object channels. Identifying cyclists from their front or back profiles is inherently challenging, sometimes even for human observers, as their slim and tall posture resembles that of pedestrians. The situation is further complicated by the presence of a nearby pedestrian which may have caused additional interference and confusion for the model.

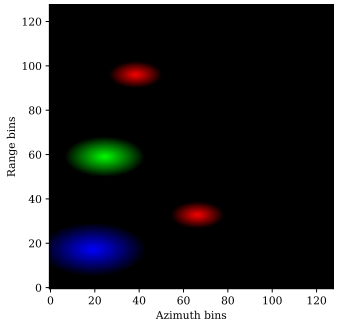
(a) Camera image



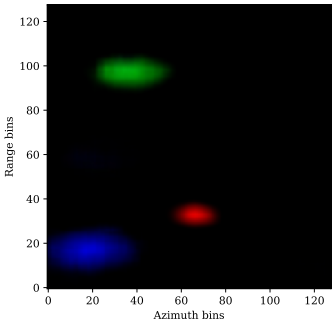
(b) Detection results



(c) Input radar spectrum



(d) Confmap ground truth

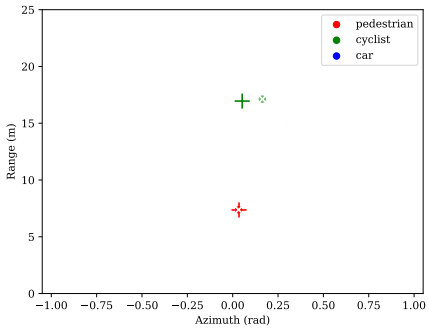


(e) Confmap prediction

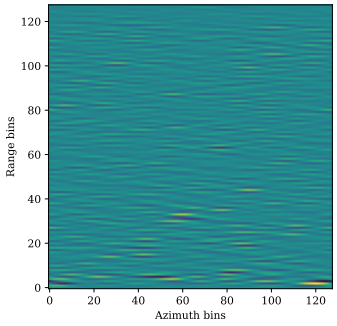
Figure 4.9: Case study of Sequence `2019_04_30_MLMS001, frame 704`. This frame is captured on the same campus road as Figure 4.8, all three object classes are present. A cylist partially occluded by the car is missed, and a pedestrian is falsely detected as a cyclist.

In Figure 4.9, we see a similar environment as in Figure 4.8, but with the presence of a car that partially occludes a cyclist. The cylist happens to be at twice the range of the car, and its radar signature is likely mixed with an echo wave of the car that bounces off the ego vehicle. As a result, the model only produces a weak response at the cyclist's location in both channels. This issue is also observed in Figure 4.7. Additionally, a pedestrian across the road is misclassified as a cyclist. It is worth noting that this pedestrian, being a different person, is at exactly the same location as the pedestrian in Figure 4.8, which was also misclassified as a cyclist. While misclassification of pedestrians and cyclists may not pose a significant safety risk in most cases, the consistent misclassification of pedestrians at this particular location raises concerns about potential systematic weaknesses in the model's receptive field at certain ranges and angles. This will be further investigated in subsection 4.6.2.

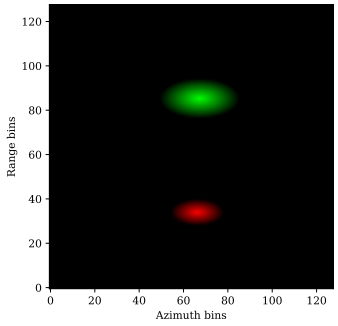
(a) Camera image



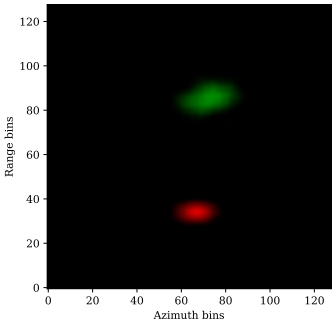
(b) Detection results



(c) Input radar spectrum



(d) Confmap ground truth



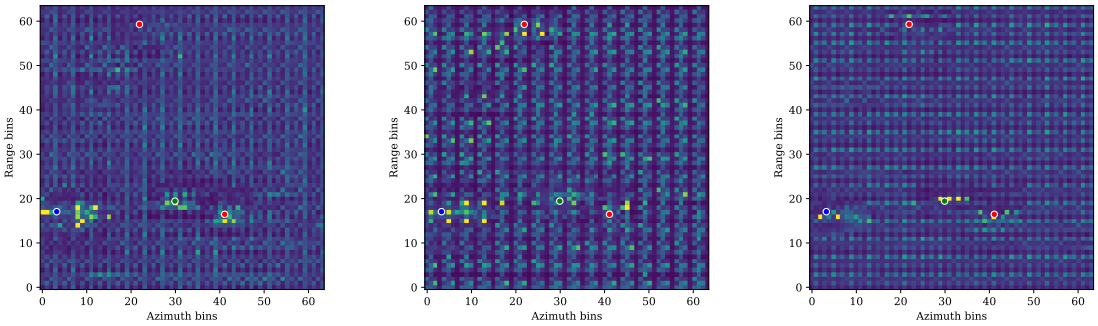
(e) Confmap prediction

Figure 4.10: Case study of Sequence `2019_05_23_PM1S013, frame 24`. This frame is captured in the same parking lot as Figure 4.7, with a pedestrian and a cyclist present. The cyclist is occluded by the pedestrian, but still detected with slight offset.

In Figure 4.10, we provide an example of successful detection in a difficult situation. The cyclist is nearly completely occluded by the pedestrian from the radar's perspective, yet the model is still able to detect both objects. The detection of the cyclist shows an irregular shaped blurry response in the output confidence map, slightly offset from the ground truth location, showing that the model is still uncertain about the exact position of the cyclist. While this result is encouraging, it also shows the limitation of the current radar system's angular resolution.

4.6.2 Attention Visualization

To further understand the model's decision-making process, we visualize the attention maps from the multi-head self-attention layers in the Transformer encoder. The attention weights



(a) Head 0 (b) Head 1 (c) Head 2

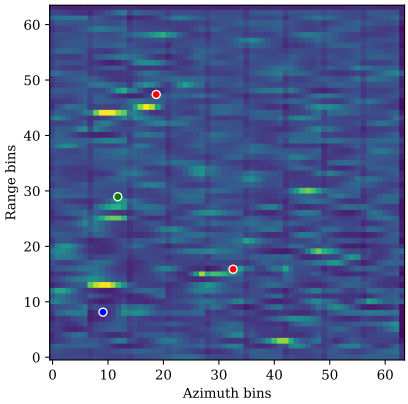
Figure 4.11: Attention map visualization. Each subplot represents the attention weights from a different head in the multi-head self-attention layer. Brighter areas indicate higher attention weights, showing where the model focuses when processing the input radar spectrum. Taken from the first block in the last stage of the decoder, when processing frame 104 of sequence `2019_04_30_MLMS001`.

($A_{w,h}$ from Algorithm 1 line 12) from each head are extracted and rearranged to match the spatial dimensions of the input spectrum and the output confidence map. An example of the attention maps is shown in Figure 4.11.

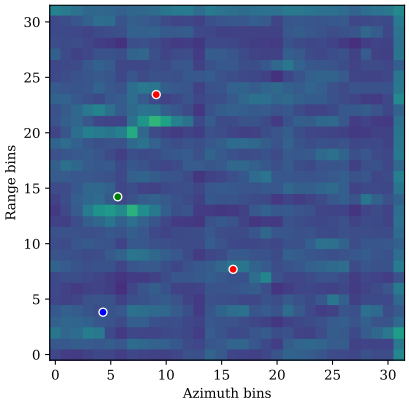
In Figure 4.11, the three heads show similar overall attention patterns with minor differences. The entire maps appear pixelized and noisy, but four distinct bright regions (although disrupted by noise) can be identified, corresponding to the locations of four true objects. Further observation reveals that the noise appears on cycles of 4 pixels in both dimensions. Considering that the features have been upscaled twice by a factor of two in the decoder, this pixelization effect likely originates from the transposed convolution layers used for upsampling. Despite the noise, it is evident that the model is able to focus on the true object locations, indicating that the self-attention mechanism effectively captures relevant features from the radar spectrum.

Understanding the role of different stages and blocks in the Swin Transformer backbone, as well as how features evolve through the network, is crucial for interpreting the model's behavior at a higher level. To this end, we visualize the attention maps from different stages and blocks in the Swin Transformer backbone, as shown in Figure 4.12.

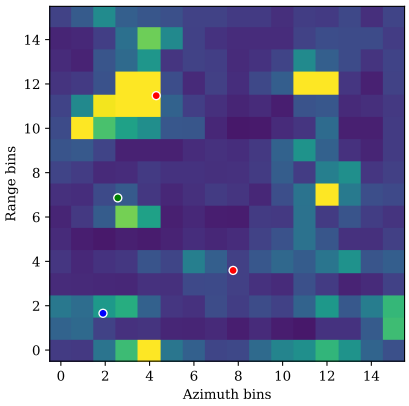
In Figure 4.12, we arrange the attention maps from different stages and blocks in the



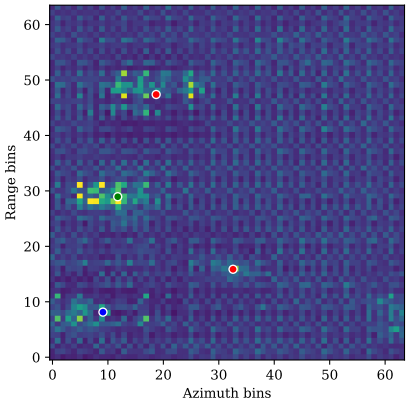
(a) Encoder stage 1, block 1



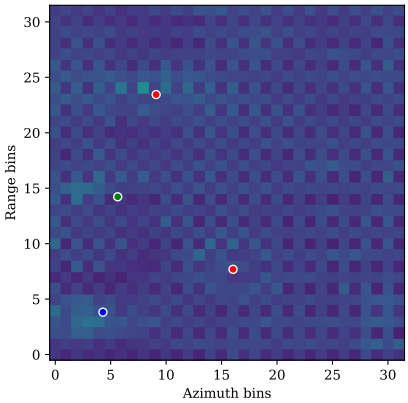
(b) Encoder stage 2, block 1



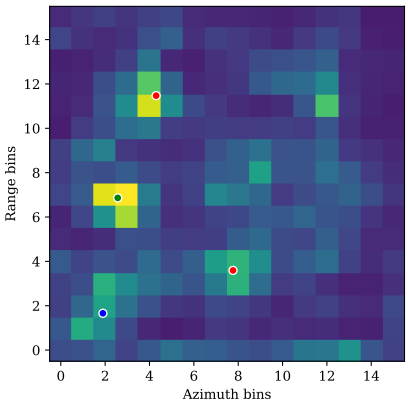
(c) Encoder stage 3, block 1



(f) Decoder stage 2, block 1



(e) Decoder stage 1, block 1



(d) Encoder stage 3, block 5

Figure 4.12: Attention visualization. Each subplot represents averaged attention weights from different heads in a Swin Transformer block, organized anticlockwise from the top-left. The same frame as in Figure 4.9 is used. Object locations are annotated with colored circles.

Swin Transformer backbone in a U-shaped layout, following the architecture of the network. Attention maps with the same spatial resolutions are placed in the same rows, with the first stage of the encoder at the top-left, and the last stage of the decoder at the top-right. Going anti-clockwise, we can observe the evolution of attention patterns through the network. Early stages of the encoder (Figure 4.12 (a) and Figure 4.12 (b)) show tendency to focus where strong responses are present in the input radar spectrum, including both true objects and noise. We also observe grid-like artifacts in the first stage (Figure 4.12 (a)) with a period of 7 pixels, which coincides with the shifting window size. Pixels on the window boundaries tend to have lower values compared to other pixels. This indicates that the shifting window mechanism in Swin Transformer may cause information loss at the window edges, which could potentially affect the model's ability to capture features on window edges. Closer inspection reveals that undetected cyclist in Figure 4.9 happens to be located on two windows boundaries (the 6th azimuth bin and the 14th range bin) in Figure 4.12 (b), adding further evidence to this hypothesis. As we progress deeper into the third stage of the encoder (Figure 4.12 (c) and Figure 4.12 (d)), we see a clear shift in attention towards the true object locations. The first block in the stage shows high values scattered across the map, while the fifth block attends more distinctly to the objects. This indicates that the third stage of the encoder plays a crucial role in refining features on a global scale and identifying true objects from noise. Moving into the decoder stages (Figure 4.12 (e) and Figure 4.12 (f)), as the features are upscaled and combined with encoder features through skip connections, we start to observe artifacts across the attention maps likely introduced by the upsampling process. While objects now appear weaker and disrupted by noise, they are still discernible and strengthened in the final stage. Five distinct bright regions can be identified in the final stage, four of which correspond to the locations of true objects, and the fifth one is likely generated by an interference outside the camera view. This proves the decoder's ability to decode object related features, but the missed detection of the occluded cyclist suggests that issues might exist in downstream modules, e.g. the MLP head.

4.7 Summary

In this chapter, we showed that Swin Transformer blocks are capable of forming a strong backbone for object detection with radar spectra. The model shows promising performance on the four test sequences from various scenarios, and achieved acceptable AP and AR for most object classes at different ranges and azimuths, though some challenges remain in azimuths with limited samples. Objects with smaller radar cross sections, e.g. pedestrians, are more adversely affected by long range and extreme azimuth angles. Case study of weak and failure examples revealed both systematic issues and incidental factors that affect the performance of proposed paradigm, such as misclassification between objects with similar radar signatures, misdetections due to multipath reflections, and challenges in occlusion scenarios. Performing object detection on the RA spectrum (polar BEV) has been proven effective with acceptable localization accuracy in most scenarios. Attention visualization provided insights into the model's decision-making process at both low-level and high-level, revealing the evolution of attention patterns through different stages and blocks in the Swin Transformer backbone. It has been observed that the model is able to focus on true object locations, with the third stage of the encoder playing a crucial role in refining features on a global scale. However, issues related to the shifting window mechanism and upsampling artifacts were also identified in shallow layers of the encoder and decoder, respectively. Information loss at window edges might have contributed to missed detections. While attention artifacts from upsampling were observed, there is no conclusive evidence that they directly lead to misdetections. On the other hand, undetected objects with high attention responses in the backbone suggest that the final MLP head might play a role in certain failure cases. In the next chapter, we will attempt to mitigate these issues through the design of improved model architectures and components.

Chapter 5

Compact Radar Object Detection

This chapter presents the design and implementation of a compact radar object detection model coined *mRadNet*, which aims to address the shortcomings of the model presented in Chapter 4. The overall architecture of mRadNet and its key designs choices are detailed, along with motivations for these choices. Experiments are conducted to evaluate the performance of mRadNet against existing methods, demonstrating its effectiveness in radar object detection tasks.

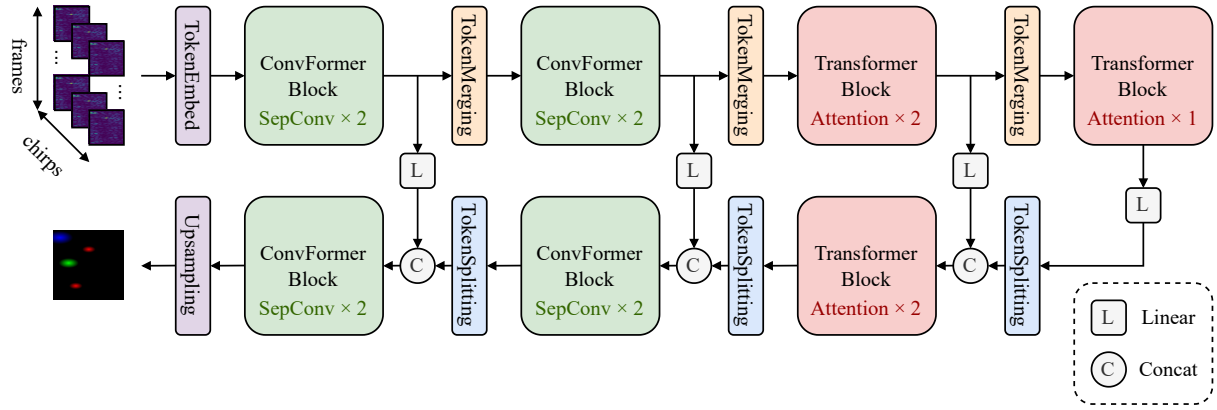


Figure 5.1: Overview of the backbone architecture.

5.1 Overall Architecture

We base our new model on the same U-Net style architecture as in the previous chapter, as shown in Figure 5.1. This hierarchical encoder-decoder design has shown to be effective through qualitative analysis in subsection 4.6.2. Therefore, we keep a similar overall architecture and focus on improving the building blocks inside the architecture. The only major difference lays in the configuration of the encoder and decoder. We added an extra stage in both the encoder and decoder to increase the model depth, which we found to be beneficial for performance.

5.2 Token Mixing

In subsection 4.6.2, the attention maps of the Swin Transformer blocks throughout the network are analyzed. While these window-based attention modules are effective at extracting multiscale features when stacked hierarchically, they are found to be causing problems at certain locations in the features, e.g., boundaries of shifting windows. Since the same window size is used throughout the network, window boundaries from different layers tend to align, leading to persistent blind spots in the attention maps, thus this mechanism is excluded from the new model.

When we look back into the history of Transformers for computer vision, we can see a constant effort in incorporating convolution's inductive biases into the attention mechanism. The original Vision Transformer (ViT) [14] sees local patches of an image as tokens. While locality exists to some extent within each token, there is no explicit modeling of local structures among neighboring tokens. To address this issue, many follow-up works [65, 66] introduce various forms of token merging mechanisms to aggregate neighboring tokens into larger tokens, thus forming a hierarchical representation that captures features at multiple scales, an idea borrowed from CNNs. Swin Transformer [2] goes one step further by restricting the attention mechanism's reception field to non-overlapping local windows for

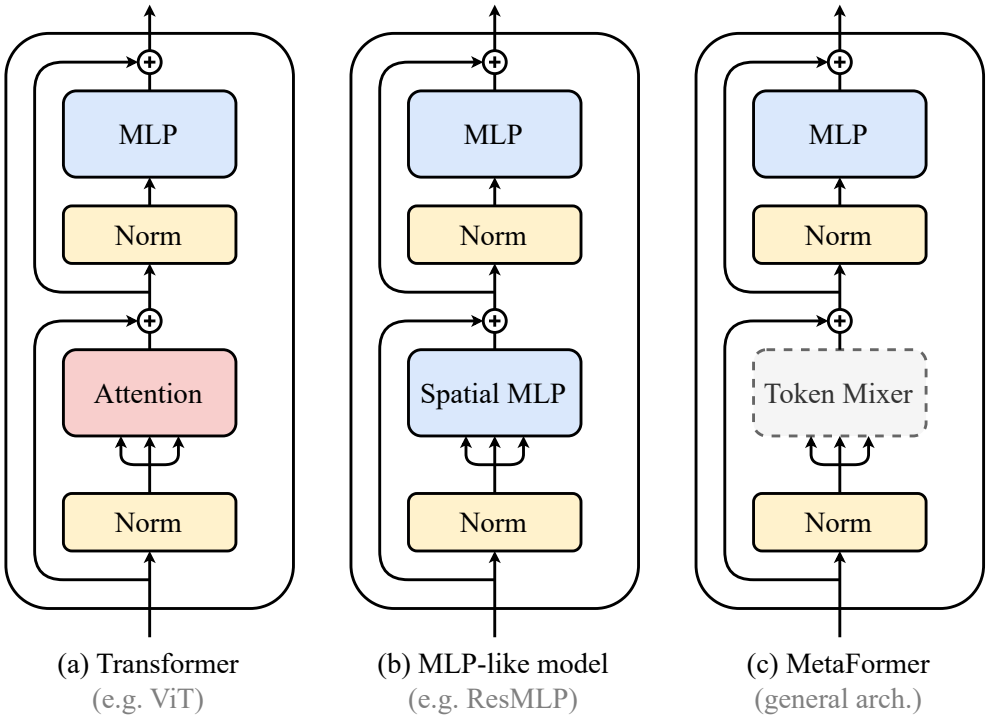


Figure 5.2: MetaFormer. MetaFormer is a generalized architecture abstracted from Transformer by not specifying the token mixer. When using Attention/MLP as the token mixer, MetaFormer reduces to Transformer/ResMLP.

computational efficiency.

Recall that Swin Transformer's window-based attention causes artifacts in the attention maps (as discussed in subsection 4.6.2), to maintain this advantage while avoiding the problems caused by windowing, MetaFormer [3] provides a new perspective by decoupling the attention mechanism from the overall Transformer architecture, as shown in Figure 5.2.

Figure 5.2 (a) shows a classic vision Transformer block, which consists of two residual blocks connected in series. The first residual block contains a MHA module with pre-normalization, while the second residual block contains a MLP module with pre-normalization. ResMLP [67] takes inspiration from this design and replaces the MHA module with a spatial MLP module, as shown in Figure 5.2 (b). The authors claim that the first residual block (a *cross-patch* linear layer) is responsible for mixing information across different spatial locations, while the second residual block (a *cross-channel* linear layer) is responsible for mixing information across different channels. Based on this observation, MetaFormer [3] steps fur-

ther by putting forward the concept of a *token mixer*, which can be any module that enables information exchange between different spatial locations, as shown in Figure 5.2 (c). The authors prove the effectiveness of this generalized architecture by demonstrating that even extremely simple token mixers, such as pooling or even random mixing, can achieve comparable performance to Transformers and CNNs on image classification tasks with smaller model sizes.

With MetaFormer ensuring solid lower bound of performance, we have gained great flexibility in the choice of token mixers, and our problem now boils down to finding a lightweight yet effective token mixer that specializes in modeling local structures, but does not introduce artifacts like window-based attention. It is also expected to fully utilize the priors of images, namely locality and translation invariance. After exploring various options, we opt for a fundamentalist approach by directly using convolution as our local token mixer. Convolution is similar to attention in many ways; both mechanisms aim to aggregate information from other spatial locations via weighted sums. Attention does so by calculating the weights dynamically based on the input features on a global scale, while convolution uses static weights on a local scale. Basically, convolution can be viewed as a form a *translation-invariant windowed attention*, where the "attention weights" are shared across different spatial locations instead of being calculated dynamically for each location. The fore-mentioned inductive biases are also naturally satisfied. Convolution comes with built-in channel mixing ability, which is redundant in the MetaFormer architecture and needs to be removed. The resulting module is depthwise separable convolution (SepConv) [68], as shown in Figure 5.3 (a).

One main difference between the three modules lies in the dimension of the kernels. In standard convolution, each kernel spans across all input channels with a size of $C_{in} \times K_h \times K_w$, where C_{in} is the number of input channels, and K_h and K_w are the kernel height and width, respectively, as shown in Figure 5.3 (a). This allows each output channel to aggregate information from all input channels within the local neighborhood, which is not needed in our case. In SepConv, each kernel only operates on its corresponding input channel with a

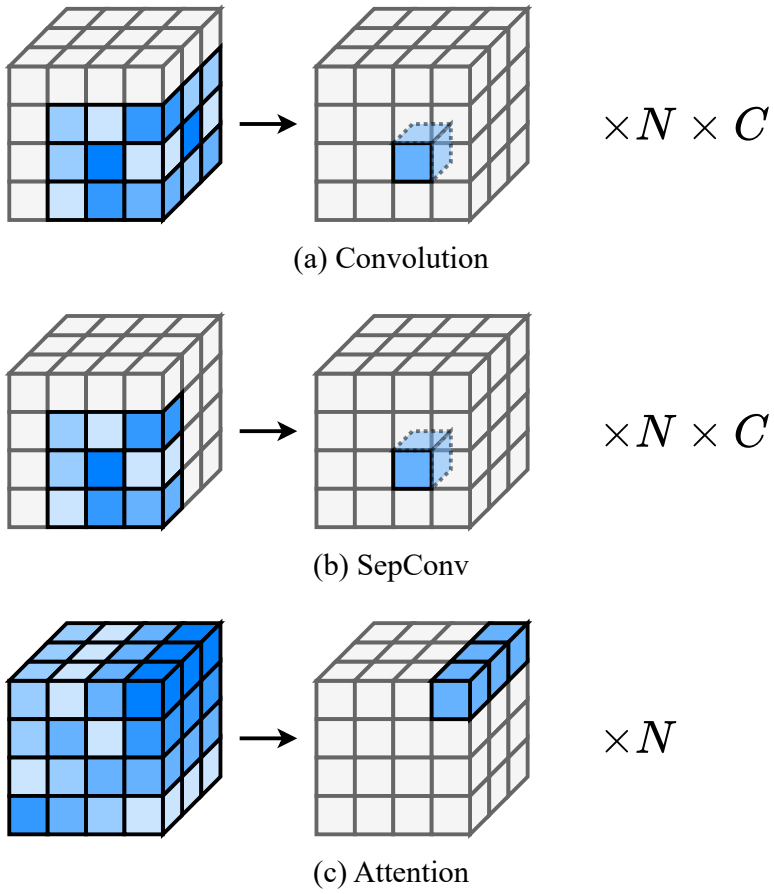


Figure 5.3: Comparison between convolution, depthwise separable convolution and attention. For each spatial location, convolution (**a**) and SepConv (**b**) operates on local neighborhoods for each channel, while attention (**c**) operates on global spatial locations across all channels. In convolution, each kernel spans across all input channels, while in SepConv, each kernel only operates on a single input channel.

Module	Time Complexity	Space Complexity
Convolution	$O(NC^2K)$	$O(C^2K)$
SepConv	$O(NCK)$	$O(CK)$
Attention	$O(N^2C)$	$O(C^2)$
Windowed attention	$O(NCK)$	$O(C^2)$

Table 5.1: Time and space complexity of different token mixers. Here, N is the number of tokens in the input, C is the token dimension, and K is the number of elements in the convolution kernel or attention window (e.g., for a 7×7 kernel/window, $K = 49$).

size of $K_h \times K_w$, as shown in Figure 5.3 (b). Attention, on the other hand, has a "kernel" with dynamic weights that spans the entire image with a size of $H \times W$, where H and W are the height and width of the input feature map respectively, as shown in Figure 5.3 (c). Another difference is that convolution and SepConv operate on each output channel independently, while attention applies the same weights across all channels.

Given an input of size $N \times C$, where N is the number of tokens and C is the number of channels, the time complexity (number of operations) and space complexity (number of parameters) of each module can be summarized as Table 5.1.

It is evident that SepConv is significantly more efficient than both convolution and attention in terms of time and space complexity, due to the removal of channel mixing and the restriction of the reception field to local neighborhoods. When compared with windowed attention, SepConv shares the same time complexity and slightly smaller space complexity, but ensures fair treatment of all spatial locations. Every token receives information from a neighborhood centered around itself, thus avoiding the window boundary blind spots mentioned in subsection 4.6.2. Note that the above discussion of SepConv as a replacement for windowed attention assumes the use of the same kernel/window size, which means that larger kernels need to be used (as opposed to the 3×3 kernels commonly used in CNNs).

To complement the local modeling ability of SepConv, we also include global attention modules at deeper stages in the network to capture long-range dependencies, as shown in Figure 5.1. The combination of SepConv and global attention as token mixers allows the

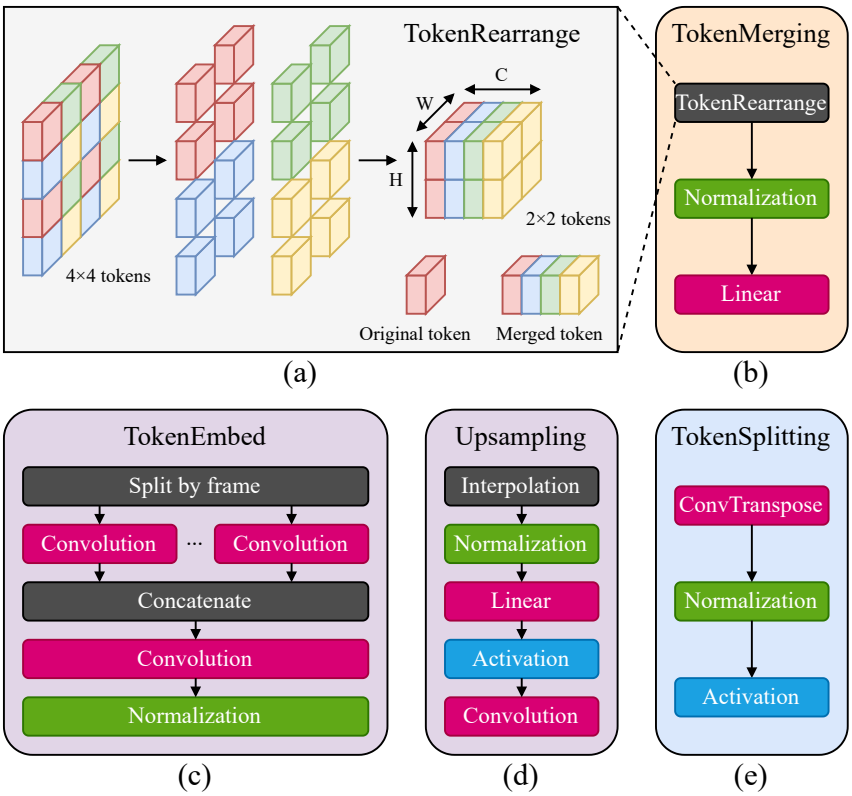


Figure 5.4: Token transformation modules. (**a**) Token Rearrange layer concatenates neighboring tokens into new tokens. (**b**) Token Merging module for downsampling. (**c**) Token Embed module for initial tokenization. (**d**) Upsampling module for final upsampling and output projection. (**e**) Token Splitting module for upsampling. In subfigures (**b**) to (**e**), operations are colored according to their categories: ■ structural layers, ■ learnable layers, ■ normalization layers, ■ activation layers.

model to effectively capture both local structures and global context, while avoiding the issues caused by window-based attention discussed in subsection 4.6.2.

5.3 Token Transformation

With MetaFormer blocks in place, our U-Net style architecture requires several additional modules to handle token encoding, decoding, merging and splitting, as shown in Figure 5.1. These modules effectively act as transition layers that connect different stages of the network, ensuring smooth flow of information and a hierarchical representation of features. Their designs are shown in Figure 5.4.

5.3.1 Token Embed

Our previous model stacked different chirps along the channel dimension to form the input tensor. While this method is straightforward and effective, it does not fully utilize the temporal relationships among chirps. To address this issue, we introduce a Token Embed module at the beginning of the network to better capture these relationships, as shown in Figure 5.4 (c). Recall that three FFTs are applied to the raw ADC data along the range, Doppler and angle dimensions respectively to obtain the RAD tensor. Since the CRUW dataset offers data in the form of complex RA tensors, speed information is implicitly embedded in the phase differences among consecutive chirps. But instead of transforming the data with an actual Doppler FFT, we opt for a learnable approach by approximating the Doppler transformation with a convolutional kernel of size $N \times 1 \times 1$ with a stride of $N \times 1 \times 1$ on the chirps, range and Doppler dimensions respectively, where N is the number of chirps. This operation merges the four chirps in every frame, and increase the number of channels by a factor of four, transforming each frame of size $B \times N \times R \times A \times C_{in}$ into a new frame of size $B \times 1 \times R \times A \times 4C_{in}$, where B is the batch size, R is the number of range bins, A is the number of azimuth bins, and C_{in} is the number of input channels. Since the radar system is inherently time-invariant, the same convolutional kernel is shared across different frames. The chirps dimension is then removed, and the resulting tensor is now an "RF video" with size $B \times T \times R \times A \times 4C_{in}$, where T is the number of frames.

At the end of the module, a 3D convolutional layer with a kernel size of $2 \times 2 \times 2$ and a stride of $2 \times 2 \times 2$ is applied for tokenizing every 8 neighboring pixels into one token, reducing the spatial and temporal dimensions by a factor of two while increasing the number of channels to the desired token dimension C.

5.3.2 Token Merging and Splitting

To construct a hierarchical architecture, each stage of the encoder and the decoder needs to be connected with transition layers that can change the spatial (and optionally temporal) resolution of the tokens, and adjust the token length accordingly. We refer to these layers as Token Merging and Token Splitting modules respectively. The designs of these two modules are shown in Figure 5.4 (b) and Figure 5.4 (e) respectively.

Token Merging modules are used in the encoder to downsample the tokens between stages. Instead of using pooling or strided convolution, we adopt a more structured approach by first rearranging neighboring tokens into new tokens with a Token Rearrange layer, as shown in Figure 5.4 (a). This operation is similar to the PixelShuffle operation commonly used in super-resolution tasks [69], but works in the opposite direction. Given an input tensor of size $B \times T \times H \times W \times C$, where H and W are the height and width of the feature map respectively, the Token Rearrange layer with a downscaling factor of 2 rearranges each non-overlapping 2×2 spatial neighborhood into a new token, resulting in an output tensor of size $B \times T \times \frac{1}{2}H \times \frac{1}{2}W \times 4C$. This operation effectively reduces the spatial dimensions by a factor of S while increasing the token dimension by the same factor, preserving all information from the input. Note that the temporal dimension remains unchanged. After the rearrangement, a normalization layer and a linear layer are applied to stabilize the training and reduce the token dimension back to $2C$, completing the downsampling process.

In the decoder, Token Splitting modules are used to upsample the tokens from each stage to a higher resolution. Naturally one would think of mirroring the downsampling process used in the encoder and reversing it. However, structurally splitting the tokens and rearranging them to different spatial locations introduces assumptions about the spatial relationships between the elements within each token, which may not hold true in practice. Therefore, we opt for a more flexible approach by using a transposed convolutional layer with a kernel size of $2 \times 2 \times 2$ and a stride of $2 \times 2 \times 2$ to directly upsample the tokens, as shown in Figure 5.4 (e). Compared with interpolation-based upsampling methods, transposed convolution is

learnable and can better adapt to the data distribution, but improper configurations may lead to checkerboard artifacts [70], which we address by carefully choosing the kernel size and stride to be equal. In addition to the upsampling operation and the subsequent linear layer, we add a normalization layer to stabilize training and avoid checkerboard artifacts seen in our previous model.

5.3.3 Final upsampling

The last stage of the decoder operates on tokens with a lower spatial resolution than the original input. Theoretically, one can simply map the tokens to a heatmap at the same resolution with a linear layer, but to minimize quantization errors and reconstruct the heatmap with smoother boundaries, we opt to upsample the tokens back to the original resolution. Instead of using another transposed convolutional layer, however, we choose to upsample the tokens with trilinear interpolation. Compared with transposed convolution, interpolation, especially trilinear interpolation, is more computationally demanding, but it is parameter-free and completely avoids checkerboard artifacts. Ensuring a smooth and artifact-free output is crucial for accurate peak detection, thus we prioritize quality over efficiency at this stage. After upsampling, the tokens are normalized, and a linear layer is applied to project the token dimension to the desired number of output channels. A final activation function is applied to map the values to confidence scores between 0 and 1.

5.3.4 Implementation Details

We train and evaluate mRadNet on the CRUW ROD2021 dataset with the same configurations as in section 4.4. The same data augmentation techniques are applied during training. Due to the introduction of the Token Embed module, however, the shape of the input tensor is slightly different. For each frame of size $4 \times 128 \times 128 \times 2$ (4 chirps, 128 range bins, 128 azimuth bins, 2 channels), the chirps are no longer stacked along the channel dimension. Instead, the input tensor retains the chirps dimension, resulting in a 4D image of shape

Table 5.2: Class-wise evaluation at different OLS thresholds.

OLS Thresh.	AP	AR	AP	AR	AP	AR	mAP	mAR
0.5	90.37%	93.85%	92.53%	96.98%	96.22%	98.71%	92.58%	96.08%
0.6	89.55%	92.92%	91.31%	95.44%	95.16%	97.77%	91.58%	94.97%
0.7	88.02%	91.01%	89.57%	93.71%	93.65%	96.62%	89.99%	93.32%
0.8	83.34%	87.43%	85.80%	90.24%	89.09%	92.74%	85.61%	89.69%
0.9	63.30%	74.24%	71.84%	79.91%	75.56%	82.53%	69.13%	78.15%
Average	84.00%	88.60%	87.13%	91.93%	90.59%	94.17%	86.70%	91.08%

$16 \times 4 \times 128 \times 128 \times 2$ (16 frames, 4 chirps, 128 range bins, 128 azimuth bins, 2 channels), which is later to be processed by the Token Embed module to a 3D image as described in subsection 5.3.1. We switch to the Smooth L1 loss function [20] for training, which is less sensitive to outliers compared with the MSE loss used in the previous chapter. The Smooth L1 loss is defined in Equation 5.1.

$$\mathcal{L}_{\text{Smooth L1}} = \begin{cases} 0.5(Y - \hat{Y})^2, & \text{if } |y - \hat{y}| < 1 \\ |Y - \hat{Y}| - 0.5, & \text{otherwise} \end{cases} \tag{5.1}$$

where $\hat{Y}$ is the predicted confidence map and Y is the ground truth confidence map. The same hyperparameters as in section 4.4 are used unless otherwise specified.

5.4 Quantitative Evaluation

The same 4 test sequences as in section 4.5 are used to evaluate the performance of mRadNet against our own previous model based on 3D Swin Transformer. Performance on each object class with different OLS thresholds are reported in Table 5.2.

Compared with Table 4.1, mRadNet achieves a slightly better overall performance than our previous 3D Transformer based model, with an increase of more than 1% in mAP. We see significant improvements in detection of cars, with more than 5% increase in AP and more than 1.5% increase in AR, making it the best performing class in this experiment.

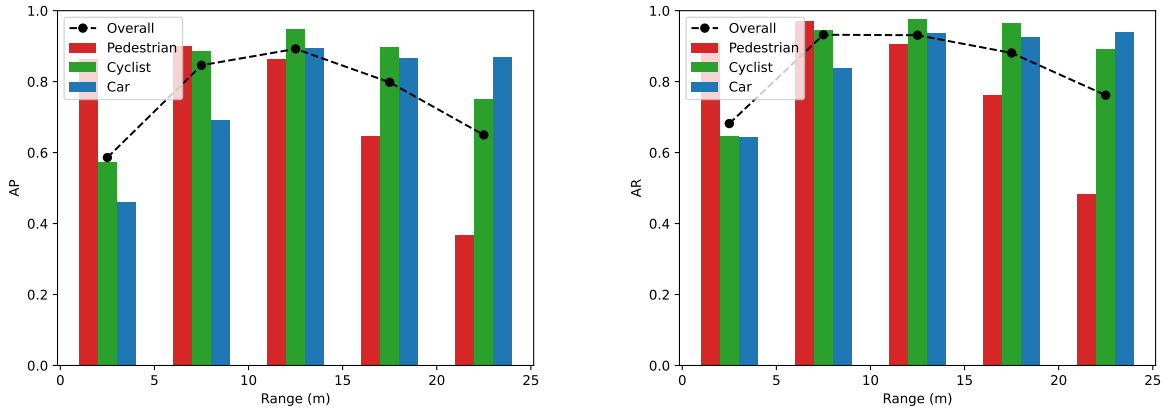
Table 5.3: Class-wise evaluation in different sequences.

Sequence	Scene	Class	AP	AR
2019_04_09_BMS1001	Parking lot	Cyclist	89.81%	91.11%
		Mean	89.81%	91.11%
2019_04_30_MLMS001	Internal road	Pedestrian	76.10%	82.10%
		Cyclist	79.77%	92.22%
		Car	92.95%	95.26%
		Mean	80.29%	87.11%
2019_05_23_PM1S013	Parking lot	Pedestrian	97.45%	98.56%
		Cyclist	98.08%	98.90%
		Mean	97.45%	98.59%
2019_09_29_ONRD005	Highway	Car	89.13%	93.50%
		Mean	89.13%	93.50%

This indicates that the proposed mRadNet is more effective in capturing the features of fast moving objects at far range, possibly due to the enhanced temporal modeling capability brought by the redesigned Token Embed module.

In line with section 4.5, we also break down the performance of mRadNet by test sequences for more detailed analysis on its strengths and weaknesses. The results are summarized in Table 5.3.

Again, we observe major improvements in car detection across all sequences, especially in the highway sequence 2019_09_29_ONRD005, in which our previous model demonstrated the worst performance, while mRadNet achieves a much higher performance that matches that of easier scenes such as parking lots. This further validates the effectiveness of the proposed mRadNet architecture in handling challenging scenarios involving fast moving objects at far range. Another noticeable change is the slight drop in cyclist detection performance in the internal road sequence 2019_04_30_MLMS001, which could be attributed to misdetections of the occluded cyclists due to more focus on learning features of cars, but more investigation through case studies and visualizations are needed to confirm the root cause.



(a) AP at different range intervals

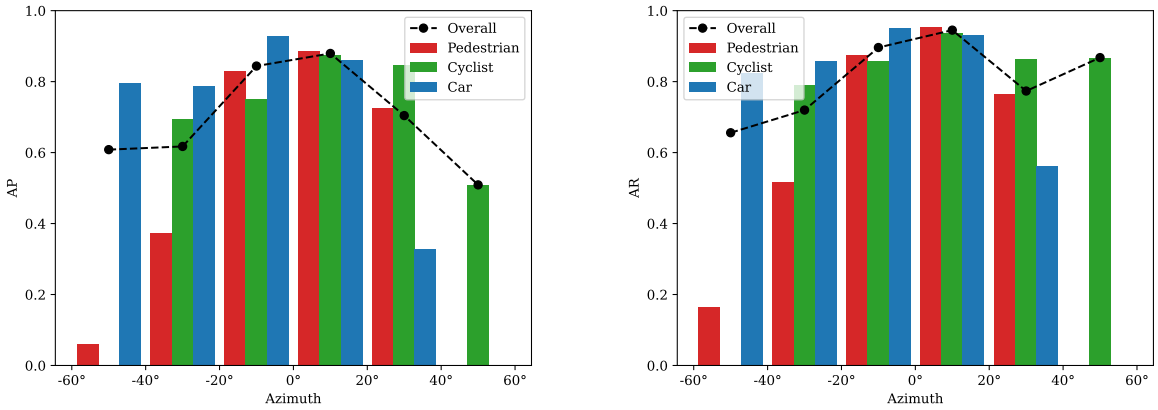
(b) AR at different range intervals

Figure 5.5: Perception performance at different ranges. AP and AR for each class are calculated for objects within different range intervals, as well as the overall mAP and mAR.

5.4.1 Range Perception

To better understand the strengths and weaknesses of mRadNet's perception capability, we further breakdown its performance at different range intervals, in line with the analysis in subsection 4.5.1. The results are illustrated in Figure 5.5.

When comparing Figure 5.5 with Figure 4.5, one can immediately notice the significant improvements in the detection accuracy at close range (0-5m). All three classes see more than 10% increase in AP in this range interval, with pedestrian detection seeing the most improvement of an astonishing 40%. However, same improvements are not seen in AR, suggesting that most of the improvements come from more precise localization of detected objects. In subsection 4.5.1, we tentatively blamed the poor performance at close range of our previous model on the limited samples at this range, but mRadNet's significant improvement suggests that precise localization can be achieved even with limited training data. Apart from improvements at close range, we also see around 10% increase in AP at far range (20-25m) for pedestrian, which was the worst performing class-range combination in our previous model. Again, this should be attributed to mRadNet's better localization capability brought by the new token mixers. Overall, mRadNet demonstrates much more balanced perception capability across different ranges, making it a more reliable choice for real-world applications.



(a) AP at different azimuth intervals

(b) AR at different azimuth intervals

Figure 5.6: Perception performance at different azimuths. AP and AR for each class are calculated for objects within different azimuth intervals, as well as the overall mAP and mAR.

5.4.2 Angular Perception

We also evaluate the perception performance of mRadNet at different azimuth intervals, following the same approach as in subsection 4.5.2. The results are illustrated in Figure 5.6.

We observe similar trends as in Figure 4.6, with mRadNet achieving the best performance at the frontal azimuth (-20° to 20°) and gradually decreasing performance towards the sides. The two models perform similarly at the frontal azimuth, but at extreme angles (beyond ±40°), mRadNet demonstrates better perception capability with around 5% higher AP for all three classes, producing a more balanced angular perception overall. Interestingly, cyclists at 40° to 60° azimuth see a huge 40% increase in AR, but the same magnitude of improvement is not observed in AP. From Figure 3.3, it can be seen that there are very few samples of cyclists at this azimuth range. While a higher AR of these rare cases does not necessarily reflect better overall detection capability, it does indicate that mRadNet is more effective in detecting distorted radar signatures of cyclists at extreme angles. This appears to be an especially desirable trait for radar perception systems, as radar resolution degrades significantly at extreme angles.

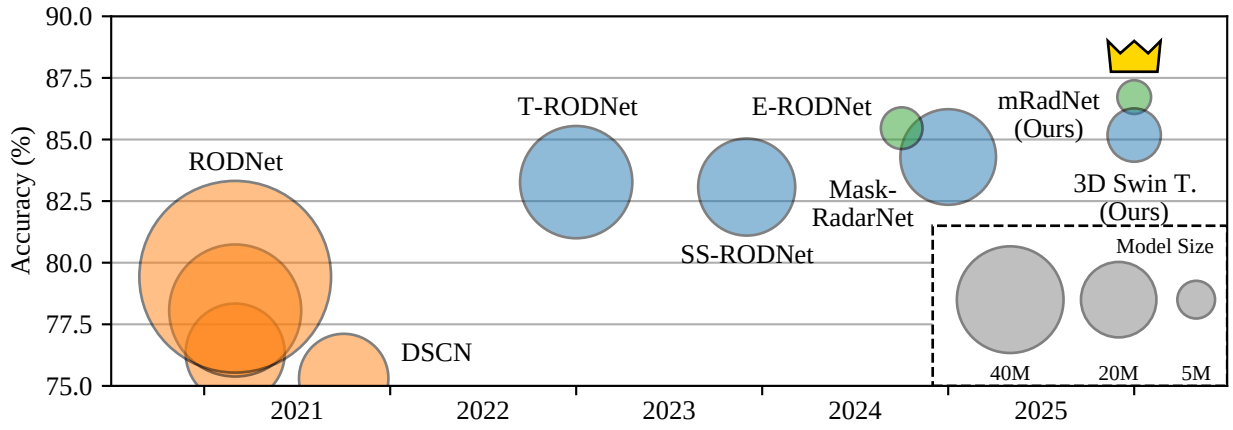


Figure 5.7: Illustrative comparison of model accuracies and sizes. ○ CNNs, ○ Transformers, and ○ MetaFormers, are represented by circles, with their sizes proportional to the model size (number of parameters) and their positions representing their year of debut and overall accuracy (mAP averaged over different OLS thresholds).

5.5 Comparative Research

To evaluate the performance of mRadNet against existing methods in the literature, we compare its results with previous State-Of-The-Art (SOTA) models that are evaluated on the same CRUW dataset. An illustrative comparison of the overall accuracy against model size is shown in Figure 5.7, and a full comparison with detailed class-wise results is summarized in Table 5.4. We also measured the model efficiency of each method in terms of number of parameters, Multiply-Accumulates (MACs), training time, and inference time, as shown in Table 5.5.

From Table 5.4, mRadNet achieves the best overall performance among all existing methods, with an mAP of 86.70% and an mAR of 91.08%, outperforming previous SOTA E-RODNet by more than 1% in both metrics. Notably, mRadNet achieves significant improvements in car detection, with more than 4.5% increase in AP compared to all other methods. This further validates the effectiveness of the proposed mRadNet architecture in the detection of fast moving objects through enhanced temporal modeling capability. For slow moving objects, however, E-RODNet still holds a slight advantage in AP for pedestrian and cyclist, although mRadNet achieves better AR for both classes. This indicates that

Table 5.4: Comparison of model performances on CRUW dataset

Model	AP	AR	AP	AR	AP	AR	mAP	mAR
RODNet-CDC[4, 5]	77.11%	79.64%	69.39%	70.02%	82.91%	89.13%	76.33%	79.28%
RODNet-HG[4, 5]	78.90%	83.81%	76.69%	78.85%	83.36%	88.55%	79.43%	83.59%
RODNet-HGWI[4, 5]	79.47%	81.85%	70.35%	71.40%	84.39%	90.05%	78.06%	81.07%
DCSN[71]	76.70%	81.50%	66.78%	69.04%	82.56%	89.52%	75.30%	79.92%
T-RODNet[43]	82.19%	85.41%	82.28%	84.30%	86.22%	92.53%	83.27%	86.98%
SS-RODNet[72]	81.37%	84.61%	83.34%	85.11%	85.55%	90.86%	83.07%	86.43%
Mask-RadarNet[73]	82.74%	85.80%	85.06%	86.67%	85.96%	90.66%	84.29%	87.36%
3D Swin T. (Ours)	83.57%	**89.85%**	87.38%	91.38%	85.34%	92.51%	85.18%	**91.02%**
E-RODNet[50]	**84.30%**	88.37%	**88.56%**	90.95%	83.87%	90.14%	85.46%	89.19%
mRadNet (Ours)	84.00%	88.60%	87.13%	**91.93%**	**90.59%**	**94.17%**	**86.70%**	**91.08%**

Table 5.5: Comparison of model efficiency on CRUW dataset.

Model	mAP	mAR	Params	MACs	Training	Inference
RODNet-CDC[4, 5]	76.33%	79.28%	34.52M	280.03G	**5h**	**1ms**
RODNet-HG[4, 5]	79.43%	83.59%	129.19M	2144.86G	35h	7ms
RODNet-HGWI[4, 5]	78.06%	81.07%	61.29M	5949.68G	177h	5ms
DCSN[71]	75.30%	79.92%	28.10M	3039.89G	113h	10ms
T-RODNet[43]	83.27%	86.98%	44.31M	182.53G	8h	20ms
SS-RODNet[72]	83.07%	86.43%	33.10M	172.80G	8h	20ms
Mask-RadarNet[73]	84.29%	87.36%	32.12M	176.91G	8h*	20ms*
3D Swin T. (Ours)	85.18%	**91.02%**	10.31M	189.16G	6h	12ms
E-RODNet[50]	85.46%	89.19%	6.10M	**33.25G**	6h	36ms
mRadNet (Ours)	**86.70%**	**91.08%**	**4.93M**	**32.79G**	**5h**	14ms

mRadNet's localization accuracy lags slightly behind E-RODNet for small and slow moving objects, which could be attributed to E-RODNet's more extensive use of convolutional token mixers that are more effective in capturing local features. Our previous 3D Swin Transformer based model also demonstrates competitive performance, achieving better overall and class-wise mAP and mAR than all existing Transformer based methods, but still falling short of E-RODNet and mRadNet.

In terms of model efficiency as shown in Table 5.5, mRadNet achieves the best performance with the smallest model size and lowest computational cost among all compared methods. We observe a trend that MetaFormer based models tend to be much smaller than both CNN and Transformer based models, while achieving better performance as well. The MetaFormer architecture empowers these models to secure a solid performance lower bound, allowing more aggressive design optimizations to reduce model size and computational cost without sacrificing much accuracy. In terms of training and inference time, CNN models such as RODNet-CDC still hold an advantage due to their shallow and largely parallelizable architectures, but their performance lags far behind more advanced models. If we rule out CNN models from the table, mRadNet also demonstrates the shortest training time and second fastest inference time among Transformer and MetaFormer based models, making it a well balanced choice in both model performance and efficiency.

5.6 Ablation Studies

To better understand the contributions of different components in mRadNet, we conduct ablation studies by removing or replacing modules in the architecture and evaluating the performance drop. Specifically, we investigate the importance of three key designs: the Token Embed module, the Token Merging module, and the SepConv token mixer. For the first two modules, we replace them with corresponding modules from our previous 3D Swin Transformer based model: stacking the chirps along the channel dimension, and convolu-

Table 5.6: Ablation study on mRadNet.

Model	AP	AR	AP	AR	AP	AR	mAP	mAR
w/o TokenEmbed	80.95%	84.97%	82.94%	91.36%	87.55%	91.25%	83.31%	88.56%
w/o TokenMerging	81.38%	84.61%	83.34%	85.11%	85.55%	90.86%	83.07%	86.43%
w/o SepConv	78.84%	85.92%	83.36%	90.44%	88.97%	92.90%	82.89%	89.14%
mRadNet	**84.00%**	**88.60%**	**87.13%**	**91.93%**	**90.59%**	**94.17%**	**86.70%**	**91.08%**

tional downsampling without normalization. For the token mixer, we replace the separable Convolution with the standard Multi-Head Self-Attention mechanism used in Transformers. The results of the ablation studies are summarized in Table 5.6.

From Table 5.6, we can see considerable performance drops in all three ablated models compared to the full mRadNet architecture. The model without separable Convolution token mixers sees the largest performance drop, with more than 3.5% decrease in mAP, indicating that the integration of convolutional token mixers plays a crucial role in enhancing the local feature extraction capability of mRadNet. This model also suffers from the most significant drop in pedestrian detection performance but sees only moderate decrease car detection performance, adding further evidence to our previous claim that precise localization of small and slow moving objects benefits greatly from convolutional feature extraction. The removal of the Token Embed module results in more than 3% decrease in mAP, with significant drops in both pedestrian and cyclist detection performance. This validates the effectiveness of the proposed learnable approach for approximating the Doppler transformation, which enhances the model's capability in capturing speed related features. Lastly, the model without the Token Merging module also experiences a notable performance drop of more than 3.5% in mAP, with significant decreases in the AR of all three classes. The checkerboard effect introduced by naive convolutional downsampling appears to have a detrimental impact on the model's ability to detect objects at certain positions, leading to more missed detections. Overall, the ablation studies confirm the importance of each key design in mRadNet, and their synergistic integration results in a powerful radar perception model that outperforms existing methods in both accuracy and efficiency.

Chapter 6

Conclusion and Future Work

6.1 Conclusion

With the development of ADAS technologies, radar sensors have seen increasing adoption in recent years due to their robustness in adverse weather conditions. Their ability to provide accurate speed information also makes them a valuable sensor for object detection tasks. However, radar object detection remains a challenging task due to the noisy and sparse nature of radar signals, as well as the lack of texture information. Traditional radar object detection pipelines often rely on hand tuned features, and lack the ability to generalize to different environments. Deep learning-based approaches have shown promising results in recent years, with mature vision architectures such as CNNs and Transformers being adapted for radar object detection. However, existing models often require large model sizes to achieve acceptable performance, leading to high inference latency and limiting their deployment in real-time applications. In this thesis, we explored two possible solutions to address these challenges. A spatio-temporal radar object detection model based on 3D Swin Transformer was proposed in chapter 4, demonstrating the effectiveness of the proposed architecture in modeling radar features and aggregating temporal information for improved detection performance. Acceptable competitive performance was achieved on various scenarios, though defects in the design of window-based attention mechanism and the upsampling module were

observed in attention visualization results, and challenges remained for in further improving the model's performance without increasing its size. To this end, we proposed a more compact radar object detection model coined *mRadNet* in chapter 5, inspired by the recently proposed MetaFormer architecture. Empowered by the flexibility of the MetaFormer architecture, mRadNet benefitted from the strengths of convolution-based token mixers and attention-based token mixers, and achieved more accurate localization while maintaining strong temporal and global feature modeling capabilities. With more efficient token encoding and decoding strategies, mRadNet achieved state-of-the-art performance on the CRUW ROD2021 dataset, with a fraction of the model size and inference time compared to existing Transformer-based models, making it a more suitable candidate for real-time applications.

6.2 Future Work

With the promising results achieved by mRadNet, several directions can be explored in future work to further improve perception performance and efficiency for ADAS applications.

While the proposed models have demonstrated decent performance on the ROD2021 dataset, the models' generalizability it is far from optimal and could benefit from further investigations into the model's generizability to different driving scenarios. In this thesis, experiments were only conducted on a single dataset with limited diversity in terms of driving environments, weather conditions, and object types. Future work couldan focus on the evaluation of the proposed models with on additional radar object detection datasets, and explore the possibility of expanding to 4D radar data for richer spatial information. Additionally, data synthesization through simulation or generative models can be investigated to augment the training data and improve the model's robustness to various scenarios.

Radar sensors are more robust in adverse weather conditions compared to emission-based sensors such as cameras and lidar, but their inherent limitations such as textureless signals and low spatial resolution still pose challenges for accurate object detection. To address these

challenges, future work can explore the integration of radar data with other high-definition sensor modalities, such as cameras and lidar, to leverage their complementary strengths. Explicit spatial representations such as BEV, as well as DETR-based end-to-end modeling approaches have shown promising results in recent years, and future work can explore the application of these techniques in integrating radar data with other sensor modalities for enhanced perception performance.

MetaFormer-based architectures have shown great potential in balancing model performance and efficiency. By decoupling token mixers from the overall architecture, MetaFormer points to a new direction for designing building blocks of computer vision models. Prior to MetaFormer, literature has largely focused on developing novel token mixers, leaving unexplored the possibilities of architectural designs that can better leverage existing token mixing techniques. Future work can focus on improving the MetaFormer architecture itself, including exploring different configurations of module arrangements within each stage, investigating alternative information flow mechanisms, and proposing designs that are more tailored to radar data characteristics.

Occupancy networks have recently gained emphasis in the field of ADAS perception due to their ability to provide dense scene understanding and class agnostic obstacle representation, which is crucial for safe navigation and decision-making. While this thesis focused on radar object detection, future work can explore the application of MetaFormer-based architecture for radar-based occupancy prediction tasks. By leveraging the strengths of MetaFormer in modeling spatial and temporal features, it is possible to develop efficient and accurate occupancy networks that can effectively utilize radar data for comprehensive scene reconstruction and obstacle detection.

Bibliography

[1] Transport Canada, "Canadian motor vehicle traffic collision statistics: 2023," Technical Report T45-3E-PDF, Transport Canada, Ottawa, ON, 2023.

[2] Z. Liu, Y. Lin, Y. Cao, H. Hu, Y. Wei, Z. Zhang, S. Lin, and B. Guo, "Swin transformer: Hierarchical vision transformer using shifted windows," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 10012–10022, 2021.

[3] W. Yu, C. Si, P. Zhou, M. Luo, Y. Zhou, J. Feng, S. Yan, and X. Wang, "Metaformer baselines for vision," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 46, no. 2, pp. 896–912, 2023.

[4] Y. Wang, Z. Jiang, X. Gao, J.-N. Hwang, G. Xing, and H. Liu, "Rodnet: Radar object detection using cross-modal supervision," in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pp. 504–513, 2021.

[5] Y. Wang, Z. Jiang, Y. Li, J.-N. Hwang, G. Xing, and H. Liu, "Rodnet: A real-time radar object detection network cross-supervised by camera-radar fused object 3d localization," *IEEE Journal of Selected Topics in Signal Processing*, vol. 15, no. 4, pp. 954–967, 2021.

[6] H. Chen, F. Hassanat, R. Laganiere, and M. Bouchard, "Mradnet: A compact radar object detector with metaformer," *arXiv preprint arXiv:2509.16223*, 2025.

[7] M. I. Skolnik, "Introduction to radar," *Radar handbook*, vol. 2, p. 21, 1962.

[8] O. Boric-Lubecke, V. M. Lubecke, A. D. Droitcour, B.-K. Park, and A. Singh, *Doppler radar physiological sensing*. John Wiley & Sons, 2015.

[9] Q. Chaudhari, "Fmcw radar part 2—velocity, angle and radar data cube." `https://wirelesspi.com/fmcw-radar-part-2-velocity-angle-and-radar-data-cube/`, November 2023. Accessed: March 2026.

[10] F. Rosenblatt, "The perceptron: a probabilistic model for information storage and organization in the brain.," *Psychological review*, vol. 65, no. 6, p. 386, 1958.

[11] P. Werbos, "Applications of advances in nonlinear sensitivity analysis," *System Modeling and Optimization*, pp. 762–770, 1982.

[12] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel, "Backpropagation applied to handwritten zip code recognition," *Neural computation*, vol. 1, no. 4, pp. 541–551, 1989.

[13] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

[14] A. Dosovitskiy, "An image is worth 16x16 words: Transformers for image recognition at scale," *arXiv preprint arXiv:2010.11929*, 2020.

[15] W. Yu, M. Luo, P. Zhou, C. Si, Y. Zhou, X. Wang, J. Feng, and S. Yan, "Metaformer is actually what you need for vision," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 10819–10829, 2022.

[16] Z. Zou, K. Chen, Z. Shi, Y. Guo, and J. Ye, "Object detection in 20 years: A survey," *Proceedings of the IEEE*, vol. 111, no. 3, pp. 257–276, 2023.

[17] Z.-Q. Zhao, P. Zheng, S.-t. Xu, and X. Wu, "Object detection with deep learning: A review," *IEEE transactions on neural networks and learning systems*, vol. 30, no. 11, pp. 3212–3232, 2019.

[18] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Advances in Neural Information Processing Systems*, vol. 25, 2012.

[19] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 580–587, 2014.

[20] R. Girshick, "Fast r-cnn," in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 1440–1448, 2015.

[21] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 6, pp. 1137–1149, 2016.

[22] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 779–788, 2016.

[23] J. Redmon and A. Farhadi, "Yolo9000: Better, faster, stronger," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 7263–7271, 2017.

[24] J. Redmon and A. Farhadi, "Yolov3: An incremental improvement," *arXiv preprint arXiv:1804.02767*, 2018.

[25] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie, "Feature pyramid networks for object detection," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2117–2125, 2017.

[26] N. Carion, F. Massa, G. Synnaeve, N. Usunier, A. Kirillov, and S. Zagoruyko, "End-to-end object detection with transformers," in *Proceedings of the European Conference on Computer Vision*, pp. 213–229, Springer, 2020.

[27] X. Zhu, W. Su, L. Lu, B. Li, X. Wang, and J. Dai, "Deformable detr: Deformable transformers for end-to-end object detection," *arXiv preprint arXiv:2010.04159*, 2020.

[28] Y. Zhao, W. Lv, S. Xu, J. Wei, G. Wang, Q. Dang, Y. Liu, and J. Chen, "Detrs beat yolos on real-time object detection," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 16965–16974, 2024.

[29] A. Ouaknine, A. Newson, J. Rebut, F. Tupin, and P. Pérez, "Carrada dataset: Camera and automotive radar with range-angle-doppler annotations," in *Proceedings of the International Conference on Pattern Recognition*, pp. 5068–5075, IEEE, 2021.

[30] M. Sheeny, E. De Pellegrin, S. Mukherjee, A. Ahrabian, S. Wang, and A. Wallace, "Radiate: A radar dataset for automotive perception in bad weather," in *Proceedings of the IEEE International Conference on Robotics and Automation*, pp. 1–7, IEEE, 2021.

[31] A. Zhang, F. E. Nowruzi, and R. Laganiere, "Raddet: Range-azimuth-doppler based radar object detection for dynamic road users," in *Proceedings of the Conference on Robots and Vision*, pp. 95–102, IEEE, 2021.

[32] J. Rebut, A. Ouaknine, W. Malik, and P. Pérez, "Raw high-definition radar for multi-task learning," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 17021–17030, 2022.

[33] K. Burnett, D. J. Yoon, Y. Wu, A. Z. Li, H. Zhang, S. Lu, J. Qian, W.-K. Tseng, A. Lambert, K. Y. Leung, *et al.*, "Boreas: A multi-season autonomous driving dataset," *The International Journal of Robotics Research*, vol. 42, no. 1-2, pp. 33–42, 2023.

[34] H. Caesar, V. Bankiti, A. H. Lang, S. Vora, V. E. Liong, Q. Xu, A. Krishnan, Y. Pan, G. Baldan, and O. Beijbom, "Nuscenes: A multimodal dataset for autonomous driving," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 11621–11631, 2020.

[35] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft coco: Common objects in context," in *Proceedings of the European Conference on Computer Vision*, pp. 740–755, Springer, 2014.

[36] M. A. Richards *et al.*, *Fundamentals of radar signal processing*, vol. 1. Mcgraw-hill New York, 2005.

[37] Z. Wei, B. Li, T. Feng, Y. Tao, and C. Zhao, "Area-based cfar target detection for automotive millimeter-wave radar," *IEEE Transactions on Vehicular Technology*, vol. 72, no. 3, pp. 2891–2906, 2022.

[38] M. Ester, H.-P. Kriegel, J. Sander, X. Xu, *et al.*, "A density-based algorithm for discovering clusters in large spatial databases with noise," in *Proceedings of the International Conference on Knowledge Discovery and Data Mining*, vol. 96, pp. 226–231, 1996.

[39] S. Im, D. Kim, H. Cheon, and J. Ryu, "Object detection and tracking system with improved dbscan clustering using radar on unmanned surface vehicle," in *Proceedings of the International Conference on Control, Automation and Systems*, pp. 868–872, IEEE, 2021.

[40] K. Fatseas, M. J. Bekooij, and W. P. Sanberg, "Optimizing pointnet++ and dbscan for object detection in automotive radar point clouds," in *Proceedings of European Radar Conference*, pp. 39–42, IEEE, 2024.

[41] A. Angelov, A. Robertson, R. Murray-Smith, and F. Fioranelli, "Practical classification of different moving targets using automotive radar and deep neural networks," *IET Radar, Sonar & Navigation*, vol. 12, no. 10, pp. 1082–1089, 2018.

[42] B. Major, D. Fontijne, A. Ansari, R. Teja Sukhavasi, R. Gowaikar, M. Hamilton, S. Lee, S. Grzechnik, and S. Subramanian, "Vehicle detection with automotive radar using deep learning on range-azimuth-doppler tensors," in *Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops*, pp. 0–0, 2019.

[43] T. Jiang, L. Zhuang, Q. An, J. Wang, K. Xiao, and A. Wang, "T-rodnet: Transformer for vehicular millimeter-wave radar object detection," *IEEE Transactions on Instrumentation and Measurement*, vol. 72, pp. 1–12, 2022.

[44] J. Giroux, M. Bouchard, and R. Laganiere, "T-fftradnet: Object detection with swin vision transformers from raw adc radar signals," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 4030–4039, 2023.

[45] A. Gu and T. Dao, "Mamba: Linear-time sequence modeling with selective state spaces," in *Proceedings of the Conference on Language Modeling*, p. 1039, 2024.

[46] L. Zhuang, Y. Yao, T. Yang, Z. Wang, and T. Zhang, "Boosting fmcw radar heat map object detection with raw adc data," *IEEE Robotics and Automation Letters*, vol. 10, no. 11, pp. 12087–12094, 2025.

[47] F. Zhang, F. Jia, and X. Zhang, "Mf-radar: A millimeter-wave radar object detection network based on frequency-enhanced mamba," in *Proceedings of Chinese Conference on Pattern Recognition and Computer Vision*, pp. 138–151, Springer, 2025.

[48] F. Jia, F. Zhang, P. Zhao, and G. Sun, "Rd mamba: A lightweight radar range-doppler spectrum object detection model," *Research Square preprint rs-6244185*, 2025.

[49] A. Sen, M. S. Mohammad, and S. Mukhopadhyay, "Ssmradnet: A sample-wise state-space framework for efficient and ultra-light radar segmentation and object detection," *arXiv preprint arXiv:2511.08769*, 2025.

[50] W. Xu, P. Lu, and Y. Zhao, "E-rodnet: Lightweight approach to object detection by vehicular millimeter-wave radar," *IEEE Sensors*, vol. 24, no. 20, pp. 33091–33100, 2024.

[51] H. Chen, K. Guo, C. Wei, Y. Zheng, H. Hu, S. Ren, and J. Liang, "Unirod: Unified radar object detection with mae-pretrained cnn-transformer fusion," in *Proceedings of the International Joint Conference on Neural Networks*, pp. 1–8, IEEE, 2025.

[52] X. Zhao, Q. Zhou, Q. Peng, Y. Xie, Y. B. Bai, Q. Wang, and R. Zhang, "Federated cnn-transformer: Enabling distributed sensing-assisted beam prediction in isac systems for iot applications," *IEEE Internet of Things Journal*, 2025.

[53] F. Al Hassanat, *Embeddable Temporal Road-User Detection From Radar: A Hybrid CNN-MetaFormer Approach*. PhD thesis, University of Ottawa, 2026.

[54] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *Proceedings of the International Conference on Medical Image Computing and Computer-Assisted Intervention*, pp. 234–241, Springer, 2015.

[55] A. Hatamizadeh, Y. Tang, V. Nath, D. Yang, A. Myronenko, B. Landman, H. R. Roth, and D. Xu, "Unetr: Transformers for 3d medical image segmentation," in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pp. 574–584, 2022.

[56] H. Peiris, M. Hayat, Z. Chen, G. Egan, and M. Harandi, "A robust volumetric transformer for accurate 3d tumor segmentation," in *Proceedings of the International Conference on Medical Image Computing and Computer-Assisted Intervention*, pp. 162–172, Springer, 2022.

[57] J. Chen, J. Mei, X. Li, Y. Lu, Q. Yu, Q. Wei, X. Luo, Y. Xie, E. Adeli, Y. Wang, *et al.*, "Transunet: Rethinking the u-net architecture design for medical image segmentation through the lens of transformers," *Medical Image Analysis*, vol. 97, p. 103280, 2024.

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

[59] J. Huang, G. Huang, Z. Zhu, Y. Ye, and D. Du, "Bevdet: High-performance multi-camera 3d object detection in bird-eye-view," *arXiv preprint arXiv:2112.11790*, 2021.

[60] Z. Li, W. Wang, H. Li, E. Xie, C. Sima, T. Lu, Q. Yu, and J. Dai, "Bevformer: Learning bird's-eye-view representation from lidar-camera via spatiotemporal transformers," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 47, pp. 2020–2036, 2024.

[61] X. Lin, T. Lin, Z. Pei, L. Huang, and Z. Su, "Sparse4d v2: Recurrent temporal fusion with sparse model," *arXiv preprint arXiv:2305.14018*, 2023.

[62] Z. Cao, T. Simon, S.-E. Wei, and Y. Sheikh, "Realtime multi-person 2d pose estimation using part affinity fields," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 7291–7299, 2017.

[63] J. Ansel, E. Yang, H. He, N. Gimelshein, A. Jain, M. Voznesensky, B. Bao, P. Bell, D. Berard, E. Burovski, *et al.*, "Pytorch 2: Faster machine learning through dynamic python bytecode transformation and graph compilation," in *Proceedings of the ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 2*, pp. 929–947, 2024.

[64] I. Loshchilov and F. Hutter, "Decoupled weight decay regularization," *arXiv preprint arXiv:1711.05101*, 2017.

[65] W. Wang, E. Xie, X. Li, D.-P. Fan, K. Song, D. Liang, T. Lu, P. Luo, and L. Shao, "Pyramid vision transformer: A versatile backbone for dense prediction without convolutions," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 568–578, 2021.

[66] L. Yuan, Y. Chen, T. Wang, W. Yu, Y. Shi, Z.-H. Jiang, F. E. Tay, J. Feng, and S. Yan, "Tokens-to-token vit: Training vision transformers from scratch on imagenet," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 558–567, 2021.

[67] H. Touvron, P. Bojanowski, M. Caron, M. Cord, A. El-Nouby, E. Grave, G. Izacard, A. Joulin, G. Synnaeve, J. Verbeek, *et al.*, "Resmlp: Feedforward networks for image classification with data-efficient training," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 45, no. 4, pp. 5314–5321, 2022.

[68] F. Chollet, "Xception: Deep learning with depthwise separable convolutions," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1251–1258, 2017.

[69] W. Shi, J. Caballero, F. Huszár, J. Totz, A. P. Aitken, R. Bishop, D. Rueckert, and Z. Wang, "Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1874–1883, 2016.

[70] A. Odena, V. Dumoulin, and C. Olah, "Deconvolution and checkerboard artifacts," *Distill*, 2016.

[71] C.-C. Hsu, C. Lee, L. Chen, M.-K. Hung, Y.-L. Lin, and X.-Y. Wang, "Efficient-rod: Efficient radar object detection based on densely connected residual network," in *Proceedings of the International Conference on Multimedia Retrieval*, pp. 526–532, 2021.

[72] L. Zhuang, T. Jiang, J. Wang, Q. An, K. Xiao, and A. Wang, "Effective mmwave radar object detection pretraining based on masked image modeling," *IEEE Sensors*, vol. 24, no. 3, pp. 3999–4010, 2023.

[73] Y. Wu, J. Liu, G. Jiang, W. Liu, D. Orlando, and L. Xiao, "Mask-radarnet: Enhancing radar object detection with spatio-temporal context," *IEEE Transactions on Intelligent Transportation Systems*, vol. 27, no. 1, pp. 1039–1051, 2025.