Object Detection with Swin Vision Transformers from Raw ADC Radar Signals

By James Giroux

Thesis Submitted in Partial Fulfillment of the Requirements for the Degree of Master of Applied Sciences in Electrical and Computer Engineering
in the School of Electrical Engineering and Computer Science
Faculty of Engineering

University of Ottawa
Ottawa, Ontario, Canada
August, 2023

Abstract

Object detection utilizing frequency modulated continuous wave radar is becoming increasingly popular in the field of autonomous vehicles. Radar does not possess the same drawbacks seen by other emission-based sensors such as LiDAR, primarily the degradation or loss of return signals due to weather conditions such as rain or snow. Thus, there is a necessity for fully autonomous systems to utilize radar sensing applications in downstream decision-making tasks, generally handled by deep learning algorithms. Commonly, three transformations have been used to form range-azimuth-Doppler cubes in which deep learning algorithms could perform object detection. This method has drawbacks, specifically the pre-processing costs associated with performing multiple Fourier Transforms and normalization. We develop a network utilizing raw radar analog-to-digital converter output capable of operating in near real-time given the removal of all pre-processing. We obtain inference time estimates one-fifth of the traditional range-Doppler pipeline, decreasing from 156 ms to 30 ms, and similar decreases in comparison to the full range-azimuth-Doppler cube. Moreover, we introduce hierarchical Swin Vision transformers to the field of radar object detection and show their capability to operate on inputs varying in pre-processing, along with different radar configurations, *i.e.*, relatively low and high numbers of transmitters and receivers. Our network increases both average recall, and mean intersection over union performance by $\sim 6-7\%$, obtaining state-of-the-art F1 scores as a result on high-definition radar. On low-definition radar, we note an increase in mean average precision of $\sim 2.5\%$ over state-of-the-art range-Doppler networks when raw analog-to-digital converter data is used, and a $\sim 5\%$ increase over networks using the full range-azimuth-Doppler cube.

Acknowledgments

First and foremost, I would like to thank my supervisors Dr. Bouchard and Dr. Laganière for providing me with this opportunity and continual feedback throughout the entire process. This work would not have been possible without their expertise and guidance. I would also like to thank NSERC and the Vector Institute for Artificial Intelligence for funding. Their generous contributions allowed me to focus solely on my research. Finally, I would like to thank my parents for their continual support in my academic endeavours and all the ups and downs that come along with it.

To my nephew, Lane.

Contents

List of Tables

List of Figures

Glossary of Abbreviations

ADC Analog to Digital Converter.

AP Average Precision.

AR Average Recall.

ASPP Atrous Spatial Pyramid Pooling.

AoA Angle of Arrival.

BEV Bird's Eye View.

CFAR Constant False Alarm Rate.

CNN Convolutional Neural Network.

DFT Discrete Fourier Transform.

DSP Digital Signal Processing.

FCN Fully Convolutional Network.

FFT Fast Fourier Transform.

FLOPs Floating Point Operations Per-second.

FMCW Frequency Modulated Continuous Wave.

FOV Field of View.

FPN Feature Pyramid Network.

GPU Graphical Processing Unit.

HD High Definition.

HPC High-Performance Computing.

IF Intermediate Frequency.

IoU Intersection over Union.

LD Low Definition.

LN Layer Norm.

MHA Multi-Head Attention.

MHSA Multi-Head Self-Attention.

MIMO Multiple Input Multiple Output.

MLP Multi-Layer Perceptron.

MUSIC MUltiple SIgnal Classification.

NLP Natural Language Processing.

NL Non-Linear.

NMS Non-Maximum Suppression.

NN Neural Network.

PA Phase Amplitude.

PC Point Cloud.

R-CNN Region Based Convolutional Neural Network.

RAD Range-Angle-Doppler.

ROI Region of Interest.

RPN Region Proposal Network.

SA Self-Attention.

SNR Signal to Noise Ratio.

SOTA State-of-the-Art.

SSD Single Shot Detectors.

SW-MHSA Shifted-Window Multi-Head Self-Attention.

ToF Time of Flight.

ViT Vision Transformer.

W-MHSA Windowed Multi-Head Self-Attention.

YOLO You Only Look Once.

mAP mean Average Precision.

mAR mean Average Recall.

Chapter 1

Introduction

1.1 Introduction

At a high level, autonomous driving systems can be broken into three sequential components. The first being sensing devices that collect information on conditions around the vehicle, of which the most common and intuitive device is a camera. The sensing devices are then coupled to information extraction algorithms, generally deep learning architectures trained to localize, and potentially classify objects within the scene. The utilization of sensor information to identify objects in a scene is generally known as *Object Detection*. Finally, the information extracted by these networks can be utilized by decision-making networks to allow autonomous navigation of the vehicle. Growing accessibility to autonomous driving systems by the general populous, via companies such as Tesla, has proved to be a double-edged sword. On one hand, it has shown that current sensing devices in the form of cameras are capable of providing information that allows autonomous agents to navigate complex scenarios. On the other, it has shown that the utilization of only one modality, such as a camera, can be inefficient and lead to hazardous behavior by the autonomous agent. The utilization of only one modality allows no possibility for corrections if the object detection network fails to identify other road users. To overcome this, research into the application of emission-based sensors such as LiDAR, and the ability of fused decision-making followed. LiDAR sensors emit continual laser pulses, reconstructing a scene based on the reflections

from a 360° scan.[1] The resulting format is a 'point cloud' (PC), where each point corresponds to a coinciding emission and reflection of a laser pulse, in a predefined cartesian coordinate system. LiDAR sensors are capable of providing accurate and detailed scene reconstruction and allow additional information to be considered in the downstream decision-making task. However, LiDAR is not without fault and suffers many of the same inadequacies of vision-based sensors given the emission wavelength. In the case of a camera, heavy snow or rain can occlude potential road hazards and result in an autonomous agent who makes unsafe decisions as a result of lens occlusion. These conditions also degrade or completely obstruct LiDAR return signal characteristics due to high levels of diffraction. As such, when conditions are optimal, camera and LiDAR sensors are adequate and preferred given their dense information content, yet for full autonomous capabilities the addition of radar must be considered.

Radar sensors operate with frequencies on the order of Gigahertz, or emit with wavelengths on the order of millimeters and are therefore unaffected by conditions such as rain or snow. However, traditional data representations of emission-based sensors are inadequate due to the loss of information via filtering. Radar sensors are subject to high amounts of noise and therefore require filtering, generally in the form of constant false alarm rate (CFAR), prior to PC formation. This creates sparse representations due to object reflection signatures being masked by noise and therefore excluded in the filtering process. The pre-processing required to form radar point clouds is computationally expensive and therefore undesirable. As a result, more traditional digital signal processing (DSP) approaches have been taken, utilizing transformed radar data that resides directly in the frequency domain via multiple fast Fourier transforms (FFTs). Commonly, three FFTs are performed creating what is known as a range-angle-Doppler (RAD) cube, where the faces of the cube correspond to range, angle, and Doppler information. Computing multiple FFTs is again computationally expensive and therefore undesirable. As such, a more desirable approach is to minimize the

[1] Generally, these pulses have wavelengths of $\sim 1000\,\text{nm}$.

number of FFTs used, in which range-Doppler spectra are suitable candidates. Utilizing only range-Doppler information one may approximate the third transformation and extract prominent object signatures to be used in downstream tasks in a single forward pass. While both the computational cost of pre-processing and the complexity of the neural network (NN) architecture decrease, one is still limited by the time needed to compute the initial transformations. Thus, the ideal model utilizes raw analog to digital converter (ADC) inputs and is capable of learning a transformation that can run optimally on a graphical processing unit (GPU) and presents the opportunity to learn characteristics of the physical radar sensor, such as the SNR (signal to noise ratio), which could be crucial given low resolution and high noise configurations.

In this work we present T-FFTRadNet, building off the prior work of Rebut et al. [1], in which we show the robustness of Vision Transformers [2] as feature extractors in object detection networks. Specifically, the capability of Swin Vision Transformers to perform object detection on high definition (HD) radar, low definition (LD) radar, and varying degrees of pre-processing, *i.e.*, raw ADC, range-Doppler matrices, and full RAD cubes. We utilize the learnable transformations from Zhao et al. [3], allowing raw ADC utilization, and show the benefits in low-resolution radar applications. Experiments are performed on the LD RADDet dataset [4] and the HD RadIal dataset [1].

1.2 Motivation

The utilization of radar in fully autonomous vehicles is crucial, but not an easy task due to the considerations mentioned prior. In this research, we aim to increase performance over existing architectures and address two main issues that currently exist with deep learning-based radar object detection networks. Namely, computational expense attributed to pre-processing, scaling network complexity relative to inputs and current network input types, and inference time which is mainly attributed to the computational complexity of performing multiple FFTs and normalization prior to injection.

As the utilization of radar becomes more widespread, the transition from high-performance computing (HPC) systems to relatively low-power hardware needs to be efficient. Network complexity (in terms of the number of parameters) will only be able to decrease to some lower bound given the complex structure of radar inputs. Instead, we focus on efficient networks with low floating point operations per-second (FLOPs) values and the ability to allocate all steps necessary for inference to a GPU, while retaining or potentially increasing performance.

1.3 Problem Statement

When dealing with the outputs produced by radar configurations in autonomous vehicles it is convenient to treat the output as an image in deep learning architectures. This is due to the high dimensional nature of said radar configurations, generally consisting of multiple receivers (R_x) and transmitters (T_x), in which each transmitter emits many times in a cyclical fashion, known as chirps. Each of these signals is then sampled by an ADC at a specific sampling rate, referred to as samples. This emission and reception scheme is known as frequency modulated continuous wave (FMCW) radar, which is discussed in detail in 2.1. The digitized output from the ADC then forms a three-dimensional tensor of shape (samples,chirps,R_x), where the number of samples and chirps is much greater than the number of receivers. Intuitively, one can then treat the receiver axis as the *RGB* channels of an image. However, our image no longer contains visual information of the vehicle's surroundings, but rather samples of pulses across each axis. In their rawest ADC form, inputs are uninterpretable to the human eye as they contain no visual cues about potential object signatures. It is from these inputs that we wish to localize and potentially classify object signatures, providing accurate and reliable range and angle coordinate estimations.

The obvious candidate for such inputs is convolutional neural networks (CNN) which have shown to be fruitful on various image-related tasks. However, CNNs possess intrinsic inductive bias which in typical image data may not always pose issues and perhaps be

beneficial. In terms of radar inputs such a bias, specifically spatial inductive bias, may pose issues given the lack of spatial context in certain representation schemes and may hinder the learning of efficient transformation estimation. Ideally, one is able to treat radar inputs as a sequence modeling network however the high dimensionality makes this impractical. We instead utilize vision transformer (ViT) networks, specifically hierarchical ViTs, treating patches of 'pixels' as sequences of tokens removing all inductive bias, employing a scheme closer to sub-sequence modeling in a practical manner, and retaining the ability to utilize the visual characteristics of traditional CNNs. Furthermore, ViTs are known for their ability to scale performance as a function of available training data. While it is true that any network will perform better given a larger training sample, the attention and embedding mechanisms of ViTs produce a more profound performance scaling.

The main contributions of this thesis are as follows:

- We utilize Vision Transformers in radar object detection applications, a first in the field.
- We show the utility of such a network on both high and low resolution radar datasets and varying degrees of pre-processing.
- We provide experimental evidence for different pre-processing schemes, such as windowing.
- We provide the first FMCW radar object detection results on raw ADC as input, utilizing complex valued linear layers.

1.4 Thesis Outline

The remainder of this thesis is organized as follows:

Chapter 2, Background: Background knowledge in related fields such as FMCW radar signal processing, the foundation of deep learning and Swin Vision Transformers.

Chapter 3, Related Works: A comprehensive literature review of image, LiDAR and radar deep learning architectures.

Chapter 4, T-FFTRadNet: Overview of our proposed deep learning architecture.

Chapter 5, Experiments: Experimental results of our architecture on both LD and HD radar datasets.

Chapter 6, Future Work And Conclusions: Summary of experimental results utilizing our proposed architecture and potential areas of improvement for future research.

Chapter 2

Background

In this chapter, we will lay the foundation for topics utilized heavily throughout the remainder of this thesis, starting with FMCW radar systems, delving into their fundamental workings and physical limitations. This will be followed by a refresher on the overall idea of deep learning and the utilization of back-propagation as a method of task learning. Finally, we will explain the concept of ViTs and their relation to hierarchical Swin ViTs, defining the key mathematical concepts used in their formulation.

2.1 Frequency Modulated Continuous Wave Radar

Radar systems operate via the transmission and reception of electromagnetic waves, in which return signals from objects are characterized by the power received. Pulsed radar systems, such as those used in the detection of aircraft, contain a singular transmitter (T_x) and receiver (R_x). The antenna is set to scan through a full solid angle radial pattern in which transmitted signals encountering objects return echoed radar pulses. Distance estimates of the echoed object are constructed using Time of Flight (ToF) information. Given that radar pulses travel at the speed of light (c), range estimates can be deduced via Eq. 2.0.1, where R is the distance between the radar system and the object, and Δt is the time delay between emission and return signal.

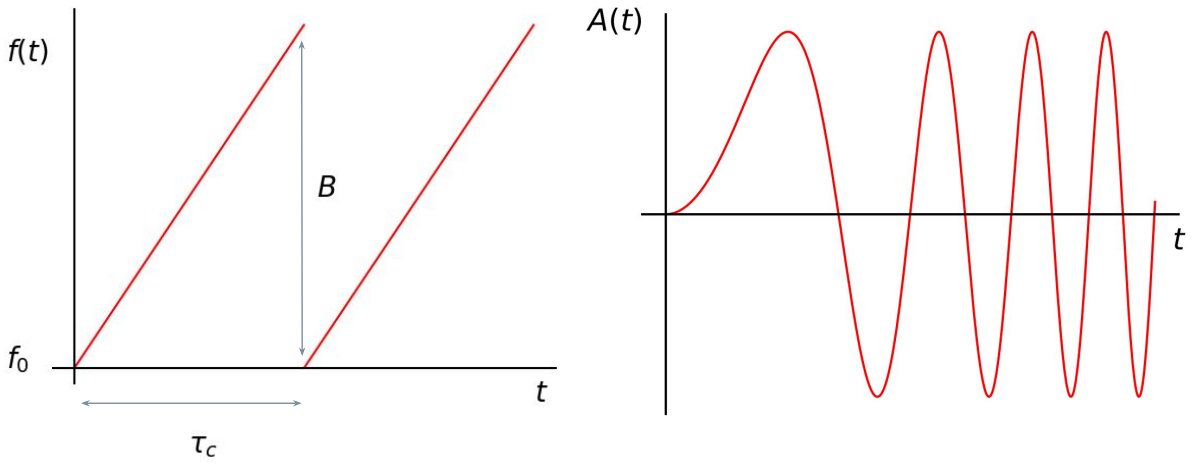


Figure 2.1: **Frequency Modulated Chirp Signals:** Two consecutive frequency modulated chirps (left). The frequency is modulated over a time period τ_c, referred to as a sweep, increasing linearly to some maximum frequency, governed by the bandwidth (B), above the starting frequency f_0. A frequency modulated sine wave utilized in FMCW radar configurations (right). The frequency scales linearly in both cases.

$$R = \frac{c \cdot \Delta t}{2} \tag{2.0.1}$$

The antenna system rotates at a specific frequency, allowing the velocity of an object to be estimated via a second detection of an object within the second revolution. This is done by utilizing the time delay of the second detection and estimating the arc length between the object's first and second positions. These two quantities allow the construction of the velocity vector. Similarly, the ϕ and θ components of the spherical coordinate position vector can be obtained via the angular values at which the antennas were set to scan when the object is detected.

Unlike traditional pulsed radar systems, modern FMCW radar systems utilize multiple T_x and R_x receivers, allowing velocity, range, and angular information to be extracted from a single scan. FMCW radar systems emit continuously, in which the output wave is frequency modulated as a function of time, Fig. 2.1 (right).

Frequency-modulated signals are referred to as 'chirped' signals, in which an individual chirp corresponds to the time τ_c needed to perform an individual sweep. A sweep is defined

as the period in which a signal is modulated from a base frequency f_0 to f'. Different frequency modulation strategies exist, in which common examples are *sawtooth*, *triangular*, and *rectangular*. Triangular modulation increases frequency linearly between an initial frequency to some maximum governed by the bandwidth (B), over one-half of the sweep. The signal frequency then decreases linearly back to the initial frequency over the second half of the sweep. In the case of rectangular modulation, the frequency follows a square wave pattern increasing acting as a step function. The radar datasets used in the remainder of this thesis utilize a sawtooth frequency modulation pattern, governed by Eq. 2.0.2. Fig. 2.1 (left) shows the sawtooth frequency modulation as a function of time for two consecutive chirps, from a singular T_x.

$$f(t) = f_0 + \frac{B}{\tau_c} \cdot t \tag{2.0.2}$$

When working with FMCW radar systems for object detection, there are three quantities one is interested in utilizing. Namely, information regarding range, velocity (Doppler), and azimuthal direction, often referred to as angle of arrival (AoA). For an emission signal with frequency f_T, phase ϕ_T and a return signal with frequency f_R, phase ϕ_R, the mixing of the two signals forms a sine wave in the form of Eq. 2.0.3.

$$A(t) = \sin\left(2\pi(f_T - f_R)t + (\phi_T - \phi_R)\right) \tag{2.0.3}$$

The mixing operation multiplies the two signals, resulting in a linear combination of frequency profiles of both the transmitted and return signal. This combination produces an intermediate frequency (IF), with constant frequency. This is represented graphically in Fig. 2.2.

The issue with FMCW radar is that the quantity Δt cannot be physically measured given that the system emits continuously. Instead, we can rely on the linearity of the emitted frequencies and utilize the inverse relationship between frequency and time domains. From

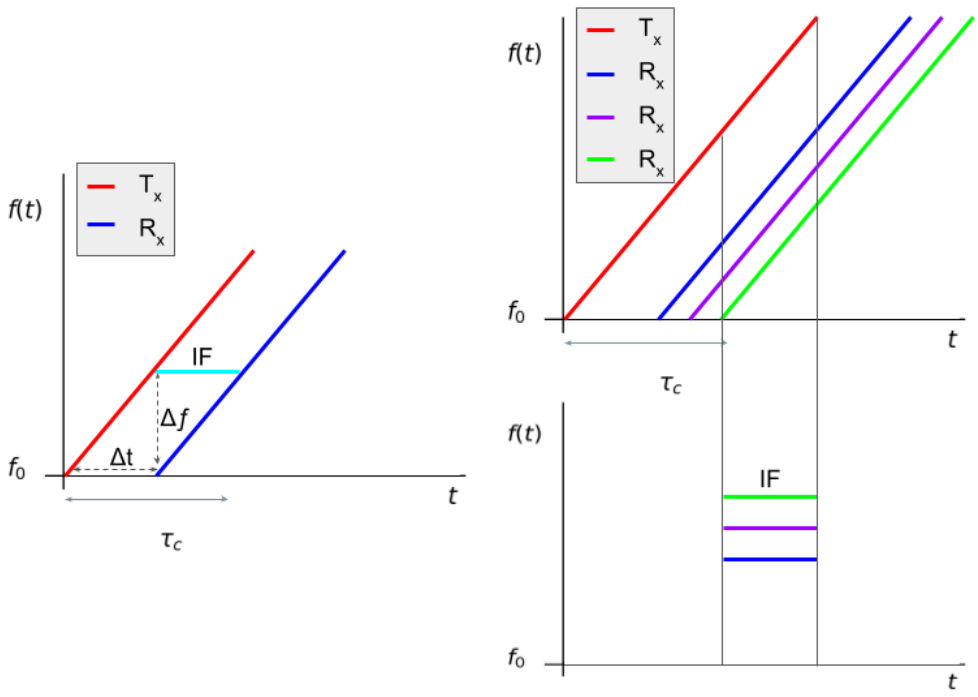


Figure 2.2: **Graphical Representation of Signal Mixing:** Singular T_x and R_x configuration. Transmitted and received signals from a singular reflected object (left) are mixed to create an Intermediate Frequency (IF). This frequency has constant value equal to the difference between the transmitted and received frequencies. The reflection of multiple objects (right, top) from an individual transmission. The objects create varying IFs in proportion to their range from the radar sensor (right, bottom). The different frequencies are easily extractable via Fast Fourier Transforms (FFTs) to produce range estimates in discrete bins based on the IF frequency response.

the above figures, we can form similar triangles to produce the relation in Eq. 2.0.4 [5].

$$\frac{\Delta t}{\Delta f} = \frac{B}{\tau_c} \implies \Delta t = \frac{\tau_c \Delta f}{B} = \frac{\Delta f}{\left(\frac{df(t)}{dt}\right)} \tag{2.0.4}$$

We can then combine Eq. 2.0.1 and the relation in Eq. 2.0.4 to formulate a range equation based on the IF response, Eq. 2.0.5.

$$R = \frac{c \cdot \Delta t}{2} = \frac{c \cdot \Delta f}{2 \cdot \frac{df(t)}{dt}} = \frac{c \cdot IF \cdot \tau_c}{2B} \tag{2.0.5}$$

In the case of multiple objects being detected, each object produces a unique IF dependent on range. We can then utilize the frequency response from an FFT to extract these

frequencies and produce range estimates. Note that the radar system will possess an inherent range resolution that is inversely proportional to the bandwidth, $\frac{c}{2B}$. In order to obtain velocity estimates of an object, one must utilize the phase difference between two consecutive sweeps. In the above formulations, we have considered only the real part of the signals. However, it is now important to consider both the real and imaginary parts of the IQ signals in order to obtain directional information from the phase shifts, where the 'I' part of the signal refers to the in-phase portion, and the 'Q' stands for the portion of the signal in Quadrature, or a signal shifted 90° out of phase. The relation of the IQ signals is given in Eq. 2.0.6, where A denotes the initial amplitude and ϕ denotes the initial phase.

$$\begin{gathered} I = A\cos\phi, \quad Q = A\sin\phi \\ Ae^{i\phi} = I + iQ \end{gathered} \tag{2.0.6}$$

Utilizing IQ signals allows the removal of ambiguity regarding the signage of output signals from the mixing of the T_x and R_x, and therefore distinguish positive from negative frequencies. Being that we treat the IQ data as complex phasors, the mixing of the transmitted (S_1) and received (S_2) signals is given by Eq. 2.0.7, in which the addition of angles translates into the addition of frequencies. Note that $\otimes$ denotes the action of an ideal mixer (multiplier).

$$S_1 \otimes S_2 = A_1 A_2 e^{i(\phi_1+\phi_2)} \tag{2.0.7}$$

For an object at distance R, moving with radial velocity v, two consecutive sweeps will produce an object with identical range peaks. The phase difference between the start of the IF signal in the first sweep is given by Eq. 2.0.8, accumulated over the time delay Δt, where λ denotes the wavelength of the transmitted signal.

$$\frac{d\phi}{dt} = 2\pi f_0 \implies \Delta\phi_1 = 2\pi f_0 \int_0^{\Delta t} dt = 2\pi f_0 \Delta t = \frac{4\pi R}{\lambda} \tag{2.0.8}$$

The second chirp measures the positional change (ΔR) of the object over one full period, which can be utilized to extract the phase difference ($\Delta\phi$) between two consecutive chirps and the velocity of said object, Eq. 2.0.9, where the direction is given by the complex component of the phase shift.

$$\Delta\phi = \frac{4\pi\Delta R}{\lambda} = \frac{4\pi v\tau_c}{\lambda} \implies v = \frac{\lambda\Delta\phi}{4\pi\tau_c} \tag{2.0.9}$$

Phase reliance introduces ambiguity into the velocity calculation given the symmetry around π and introduces an upper bound for velocity, $\frac{\lambda}{4\tau_c}$, by taking the limit of the above expression. In the case of multiple objects at the same range, but possibly different angles and speeds, utilizing only two chirps is insufficient and one instead must utilize N chirps, where N is equal to 64 and 256 in later sections. The range FFT will produce N peaks at the same location, in which a second FFT, referred to as the Doppler-FFT can resolve individual objects' frequency components given their differing phase contributions, which can then be translated into radial velocities. The angular frequency peaks may only be resolved via an FFT given their separation is greater than $\frac{2\pi}{N}$, or $\frac{\lambda}{2N\tau_c}$ in units of m/s. This leads to a velocity resolution inversely proportional to the number of chirps and their duration [6]. In order to obtain angular information we must rely on the physical characteristics of the sensor, namely the spacing of the receivers in relation to one another. The antennas are spaced in such a way that angular information is encoded into the phase delays attributed to the path length difference of received signals. Note that in this configuration we assume a far-field source in our approximations making calculations much simpler. Fig. 2.3 depicts an FMCW radar configuration in which an emitted wave is reflected from a distant object and the signal is received across two antennas with spacing $\ell = \frac{\lambda}{2}$. This spacing is common as it provides the largest angular field of view (FOV) [6].

The path length received on each antenna differs by a small amount (Δd), proportional to the phase shift, which will be extractable in the frequency domain in the form of a phase delay, given by Eq. 2.0.10.

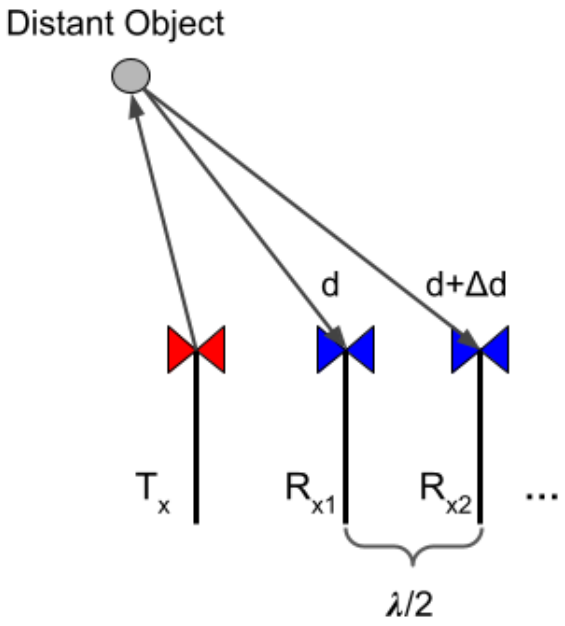


Figure 2.3: **FMCW Radar Antenna Configuration for Angular Information Extraction:** Angular information of an object's position is encoded into the differing path lengths across the receivers. These path lengths translate into phase delays in the frequency spectrum extracted by a Fast Fourier Transform (FFT), which then translate to angles from the sensors via a sinusoidal relation. Note that the angular resolution is not linear and follows the same sinusoidal relation. The physical spacing of antennas is crucial given the relation to the maximum Field of View (FOV) of the radar sensor. This is commonly chosen as one half the emitted wavelength giving the maximum FOV of $\pm\frac{\pi}{2}$.

$$\Delta\phi = \frac{2\pi\Delta d}{\lambda} \tag{2.0.10}$$

The path length difference will also be given via a trigonometric relation in Eq. 2.0.11, which can be inverted to obtain the angle (θ) at which the object is located.

$$\Delta d = \ell \sin(\theta) \implies \theta = \sin^{-1}(\frac{\lambda\Delta\phi}{2\pi\ell}) \tag{2.0.11}$$

As with Doppler information, the phase dependence introduces ambiguity into angles about $\pm\pi$. We take the limit of θ as it approaches this value to obtain the maximum unambiguous angle in Eq. 2.0.12. In our case, we have chosen the physical separation to be one-half wavelength providing a maximum FOV of $\pm\frac{\pi}{2}$.

$$\theta_{max} = \sin^{-1}(\frac{\lambda}{2\ell}) \implies \theta_{max} = \sin^{-1}(1) = \frac{\pi}{2} \tag{2.0.12}$$

Note that the angular resolution of the sensor is non-linear as a sinusoidal function, creating larger side lobe profiles in the frequency domain at larger values of θ, resulting in the potential accumulation of objects under the same frequency peaks.

2.2 Deep Learning

Deep learning has steadily worked its way to the forefront of autonomous systems over the last decade, given its ability to learn and execute a multitude of tasks. Deep learning algorithms attempt to mimic the structure of the human brain in which connections between neurons (neurons are mathematical functions in our case) are created and then weighted dependent on their contribution to the overall task. This is accomplished by repetitive exposure to experiences surrounding the task, in which connections are reformulated and weighted given the response of the algorithm. Deep learning instantiates a function, which is parameterized by a Neural Network (NN). In its simplest form, NN is a collection of weight matrices where an input vector $\mathbf{x}, x_n \in \mathbf{x}$ is modified via matrix multiplication linearly throughout the network to produce an output $\mathbf{y}$. This function can be arbitrarily defined as Eq. 2.0.13, where W_n and B are the weights and bias, respectively.

$$\mathbf{y} = f(\mathbf{x}, W_n, ..., W_0, B) = B + x_0 W_0 + x_1 W_1 + ... + W_n x_n \tag{2.0.13}$$

The above equation represents the simplest form of NN, referred to as a *Perceptron*, consisting of an individual layer (or weight matrix), producing a linear function mapping $x \rightarrow y$. In modern deep learning applications, simple linear functions are inadequate and require multiple layers with the introduction of non-linear functions between them (h), Eq. 2.0.14, where Θ denotes the set of weight matrices.

$$f(\mathbf{x}, \Theta) = h_n(W_n \cdot h_{n-1}(W_{n-1} \cdots h_1(W_1\mathbf{x}))),\ W_n \in \Theta \tag{2.0.14}$$

The method of task learning in NNs utilizes *backpropagation*, stemming from simple function optimization techniques. For observable elements $\mathbf{x}_k \in D, D \subset \mathbb{R}^n$, the optimal function f is one that minimizes the error between predicted and expected values for all $\mathbf{x}_k \in D$. In order to measure the error, we define a loss or cost function, providing tractable connections between input and output. The goal is to then minimize the cost function for the entire set D. This is done by taking derivatives of the evaluated cost function with respect to the set of weights and updating the weights via a backward pass, hence the name backpropagation. Consider the simple case of a scalar-valued cost function $\mathcal{L}(x; M)$, where M is a $k \times k$ weight matrix. Without a formal definition of the cost function itself, we can formulate the weight adjustments to be utilized in back-propagation at iteration ℓ, where λ denotes the scale factor known as the learning rate, Eq. 2.0.15.

$$M^{\ell} = M^{\ell-1} - \frac{\lambda}{N} \sum_i \frac{\partial \mathcal{L}(x_i; M)}{\partial M} \tag{2.0.15}$$

Note that it is common to utilize *mini-batches* in the gradient summation in which we utilize subsets of the dataset D to perform many updates over the course of one training iteration. This increases the efficiency at which the model is capable of learning a task. In the case of simple functions, such as linear matrix multiplication, the derivative calculation is trivial. As the complexity of the NN grows and functions become nested, the back-propagation scheme remains the same with the utilization of the chain rule. Consider now that our cost function $\mathcal{L}(x; M)$ contains multiple nested functions $\omega(u(z(\nu(x; M))))$ coupled to the weight matrix M, resulting in a gradient in the form of Eq. 2.0.16.

$$\frac{\partial \mathcal{L}}{\partial M} = \frac{\partial \mathcal{L}}{\partial \omega} \frac{\partial \omega}{\partial z} \frac{\partial z}{\partial \nu} \frac{\partial \nu}{\partial M} \tag{2.0.16}$$

Thus, the back-propagation scheme remains simplistic for even complex NN structures.

This complexity is further reduced in common deep learning packages as *auto-diff* handles all gradient calculations for the user on the backend. Furthermore, these calculations are optimized to be run in parallel on GPUs resulting in fast and efficient training schemes. Since the advent of the Perceptron and its predecessor, the Multi-Layer Perceptron (MLP), new areas of deep learning have emerged following a similar pattern of mimicking human interpretation. For example, natural language processing (NLP) models are able to convert the contextual relation between words into mathematical representations allowing deep learning algorithms to efficiently identify the sentiment and objective of sentences. At a high level, this can be thought of as forming similarity metrics via dot-product between numerical word representations, referred to as word embeddings, in order to allow the model to numerically interpret and optimize itself with respect to the objective. Arguably, the most prominent advancement in NNs was the adaptation of human vision characteristics via the utilization of windowed convolutions, forming CNNs. As the name suggests, CNNs utilize the convolution operation across a neighborhood of points within a subspace (window) of an image. The utilization of convolutional windows on grid-like data, such as images, has proven to be fruitful across a wide variety of tasks, capable of learning spatial features at varying resolutions [7]. The convolution operation is defined in Eq. 2.0.17, where κ denotes the $i \times j$ kernel and H denotes the input image.

$$(H * \kappa)[u, \nu] = \sum_i \sum_j \kappa[i, j] \cdot H[u - i, \nu - j] \tag{2.0.17}$$

CNNs utilize multiple kernels, producing multiple feature maps each with distinct information characteristics. For a given task, not all extracted features are of equal importance and we, therefore, want to filter out information only pertinent to the task at hand prior to the final processing stages. This is done utilizing pooling layers, in which a common form of pooling is to take the maximum value in subwindows of the output feature maps. This alleviates the increased complexity of network components downstream. The idea being that in a local subset of the feature map, the corresponding features will pertain to the same

information and therefore we can rely only on the maximum without a loss of generality while downsampling the output. This downsampled output can then be processed by fully connected layers, often referred to as linear layers, in order to produce the task-dependent output. Examples of which include regression labels for classification, or pixel locations for boxes containing an object within the image. CNNs have also proven to be fruitful on other input types where the lateral relation is not spatially dependent, but time dependent. In this case, one-dimensional convolutions are capable of extracting feature information from inputs such as signal amplitudes, while retaining the relationships across the time domain.

2.3 Vision Transformers

Vision Transformers [8] (ViT) leverage on powerful NLP backbones, transformers, and convert their implementation to instead operate on structured grid data such as images. CNNs operate on local sub-patches of image inputs and as such the relation between distant sub-patches can falter. ViTs instead convert input images into a series of $n \times n$ patches, each of which is flattened into a sequence and assigned a positional encoding. The positional encoding is essentially an index for a lookup table containing flattened image patches, but can also be interpreted as a spatial location index when patches are processed within the transformer. As with NLP models, we consider each sequence as a token, in which the aim is to extract all pertinent information from each patch token and combine it into a single token (referred to in the literature as the 'CLS' token) for downstream tasks.[1] An individual patch token is responsible for learning a concise representation of the image patch it is assigned to [9]. Fig. 2.4 depicts the overall network structure of a ViT designed for image classification. Note that the objective of ViTs can range from classification to bounding box regression such as with CNNs.

ViTs provide efficient information extraction via the usage of Multi-Head Attention (MHA) mechanisms, essentially training the model where to 'look' via a weighting scheme.

[1] CLS stands for classification, and is a summarization of the input image.

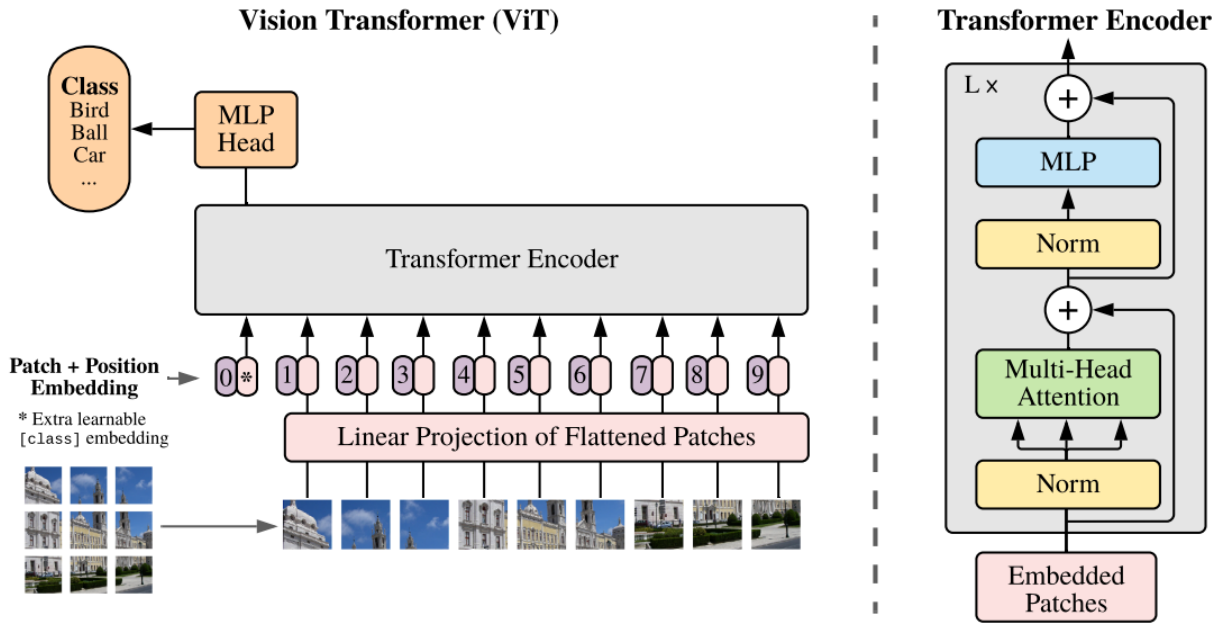


Figure 2.4: **Vision Transformer:** Vision Transformers (ViT) transform input images into a sequences of $n \times n$ patches, where each patch is assigned a positional encoding. Positional encodings are index values to be utilized in a lookup table for further processing within the transformer. The transformer encoder utilizes a series of normalization, attention and fully connected layers with the goal of extracting relevant information and encoding it into a compact representation. Image taken from [8].

Attention mechanisms utilize three component vectors, namely a *Query*, *Key*, and *Value*. Each of these vectors are feature maps, in which the Query vectors Q describe what information the model wants to attend to given a specific task, the Key vectors K describe the information content of each token and its relation to others, *i.e.*, when it should be considered and the Value vectors V are the feature map vectors in which the attention matrix is applied, producing suppression of low importance values [10]. The attention function is described mathematically in Eq. 2.0.18, where the inputs to the softmax function are normalized by the square root of the dimensionality of the corresponding Key vectors (d_K).

$$Attention(Q, K, V) = softmax(\frac{QK^T}{\sqrt{d_K}}) \cdot V \tag{2.0.18}$$

A more generalizable method of attention is known as *Self-Attention* (SA). At a high level, SA allows a sequence to learn information about itself, *i.e.*, word context in NLP or spatial

context in ViT, along with the information relative to the output task. The formulation for SA remains the same as above, although now we can think of attention as extracting not only information between multiple sequences but all combinatorics of sequences including sequences and themselves. The argument of the softmax function computes the similarity between all combinations of the Q and K matrices via a dot-product. Being that we represent each image patch as a sequence (vector) we can form a vector space with an orthonormal basis in which the angle between two vectors (dot-product) defines the relation of information content between them. The issue with the form of attention above is that for a given sequence there could be multiple sub-sequences that the model may prefer to attend to. The attention mechanism may mask the sub-sequence relations given that the argument of the softmax is a weighted average over each sequence. To overcome this, Multi-Head Self-Attention (MHSA) is used in which the Query, Key, and Value vectors are divided into sub-vectors which are then passed through the attention mechanism independently and concatenated to form the final attention output, Eq. 2.0.19, where H_i represents the output of attention head i, W_i^{α} represent learnable weight matrices for the sub-vector attention and W^0 is a learnable matrix.

$$\begin{aligned} \text{MHSA}(Q, K, V) &= Concat(H_1, ..., H_n) \cdot W^0 \\ H_i &= softmax(Q_i W_i^Q (K_i W_i^K)^T) \cdot W_i^V V_i \end{aligned} \tag{2.0.19}$$

As shown in Fig. 2.4 (right), the output of the MHA (or MHSA) is processed via a normalization layer and an MLP before being concatenated with the original MHA output. This output scheme forms the transformer encoder producing a fixed-length feature vector to be used in downstream tasks, such as image classification via an MLP (left). ViTs for usage in feature extraction, such as Swin Vision Transformers, operate in a very similar fashion to the process discussed above, although the utilization of a singular token for information extraction is disregarded. Instead, individual tokens are retained and merged given their index position in order to retain grid-like structures for consecutive encoder blocks. More

information on Swin Vision Transformers is provided in Chapter 4.

Chapter 3

Related Works

3.1 Object Detection and Segmentation using Images

Over the last decade, deep learning as the primary method of object detection and instance segmentation has been dominant in the space of image data. In the space of image object detection there are three dominant algorithm types:

- Region Based Convolutional Neural Networks (R-CNN)
- You Only Look Once (YOLO)
- Single Shot Detectors (SSD)

Generally, image-based object detection algorithms can be broken into three components: the backbone, the neck, and the detection head. While the task remains constant, the individual network development differs in methodology. In what follows we dissect and analyze the three dominant image object detection networks identifying crucial components, along with the advantages and disadvantages of each differing network configuration.

3.1.1 Region Based Convolutional Neural Networks

Algorithms such as Faster Region-Based Convolutional Neural Network (R-CNN) [11], have allowed near real-time object detection with inference times on the order of hundreds of

milliseconds, building off the predecessors in [12, 13]. R-CNN architectures generally consist of 4 components, a Region Proposal Network (RPN), a backbone network, the neck (or deeper feature extraction layers), and classification heads. In order to efficiently identify objects in an image, an RPN first extracts regions of interest (ROIs) to be fed to feature extraction backbones. This allows the backbone to extract prominent features of only regions considered information-dense, *i.e.*, regions that contain objects of interest. Initial implementations of R-CNN generated a fixed number of ROIs, but have since been improved to allow the unification of both the RPN and feature extraction networks, where at a high level the RPN acts as an attention mechanism over the feature maps [11]. The feature maps are then fed to the classification and bounding box heads to localize and categorize objects within the ROIs. These algorithms revolutionized the space of image-based object detection and laid the groundwork for many implementations to follow, such as Mask R-CNN [14]. Mask R-CNN provides the capability to produce full segmentation maps of injections in parallel to bounding box predictions via the addition of a third head. Segmentation is key in the space of autonomous vehicles, given that we do not only want to classify potential objects in the scene, but also provide information on potential driving surfaces for the vehicle to operate, or provide a more fine-grained representation of objects present. Object detection on images usually consists of either 2D or 3D boxes, where an injected image produces a set of pixel coordinates $(\mu_1, ..., \mu_n, \nu_1, ..., \nu_n)$ via a forward pass through a NN. Given a known projection matrix P, it is then possible to reproject these pixel coordinates to Cartesian representations allowing geometrical interpretations by downstream tasks. Segmentation instead aims to classify objects on a pixel by pixel basis, producing probability maps of size (M, N, C), where M, N, and C represent the number of pixels in the image and classes. Each channel in the output contains probabilities of corresponding classes which can be utilized to classify objects at a pixel level allowing finer-grained object localization over traditional bounding box predictions.

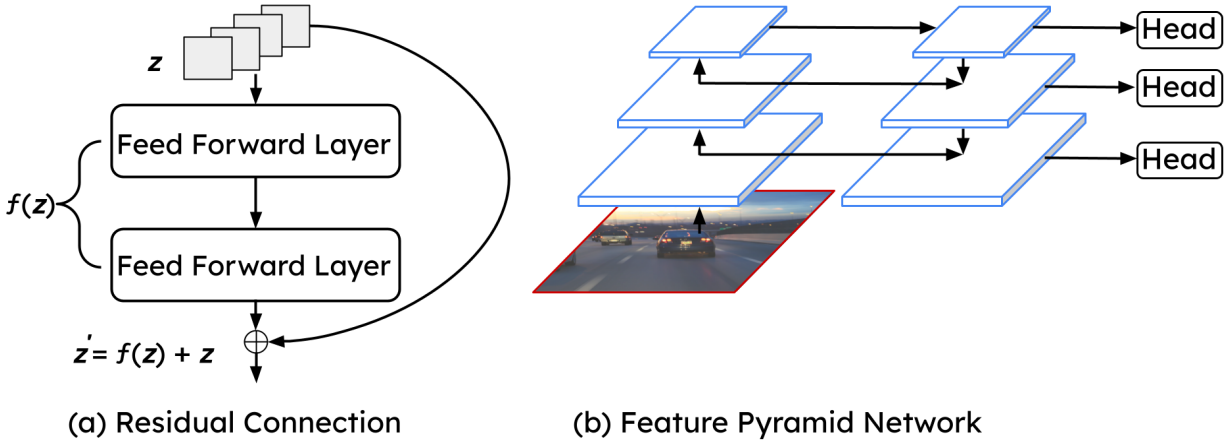


Figure 3.1: **Residual Connections:** A residual connection *(a)*, combines upstream information with downstream features via a concatenation or addition operation. We process our input z via a function f parameterized by a Neural Network (NN). This idea is consistent with Feature Pyramid Networks (FPN) *(b)*. An FPN extracts features of varying resolution via convolutions in a downscaling operation, which are then connected laterally to the upscaling portion of the algorithm via a concatenation or addition.

Architectures such as ResNet [15] and its many variants have also helped mold the network structure utilized in object detection tasks and commonly form the backbones in networks such as Faster R-CNNs. Resnets provide increased performance over more traditional backbones such as VGG-16 [16] and AlexNet [17] seen in the preceding architectures such as R-CNN. Resnet structures utilize 'residual connections', sometimes referred to as 'skip' connections, allowing the aggregation of upstream features with downstream outputs. Fig. 3.1*(a)* depicts an example of a residual connection where we parameterize our function f via a NN.

Utilization of skip connections allows higher-level feature representations to be retained in deeper portions of the network and promotes more efficient feature extraction. The idea of connecting high and low-level features is common in other networks such as Feature Pyramid Networks (FPN) [18], Fig. 3.1*(b)* where top-down projections and connected laterally via concatenation operations. The top-down structure of FPNs allows features of varying resolution to be extracted from an image and reprojected into the output feature maps via the later connections at the same resolution. FPNs offer the capability of performing object

detection via bounding box proposal when coupled to RPN networks and end-to-end object detection, in which they form the backbone of algorithms such as Faster R-CNN coupled to ResNet implementations [19].

3.1.2 You Only Look Once

YOLO [20] detectors offer the fastest inference speed at the expense of significant localization errors. YOLO networks allow the production of bounding boxes and classes in one forward pass, without the need for dedicated detection heads. At inference, input images are divided into evenly spaced grids of size $M \times M$, and for each grid, the model produces B bounding box predictions where each bounding box has an associated confidence and C class probabilities. The resulting output contains N, $M \times M \times (B \cdot 5 + C)$ tensors, where N is the number of grid cells, and 5 corresponds to the pixel coordinates (μ, ν), bounding box width and height (w, h) and confidence C [20]. The network is able to aggregate features from all locations in the image in order to regress bounding boxes, introducing a level of global reasoning to the detection network, as opposed to networks such as R-CNN, which utilize only features from selected ROIs and are therefore limited by the quality of the RPN. Moreover, it does not require pre-training of a separate RPN as the inference objectives remain constant throughout all network components. YOLO has undergone significant updates over the years, in which a notable change was introduced with YOLOv2 [21]. YOLOv2 introduced the utilization of anchor boxes, which are defined as a predetermined set of boxes suitable to contain object candidates derived utilizing object distributions. These boxes are generally over-compensations for true object size and are refined in the prediction head of the networks to better encapsulate the true objects' image representation. The model does not need to predict bounding boxes directly, but rather to localize the center of anchor boxes given a class hypothesis and provide corrections to provide better bounding box dimensions. This greatly improved the localization error issue seen with the initial implementation of YOLO. Since then, many iterations v3-v5 [22, 23] have emerged offering performance increases over

previous architectures.[1] The most notable being the introduction of the Auto-Anchor box algorithm which automizes the process of deriving anchor boxes. This is crucial given that different cameras possess different aspect ratios and therefore object bounding boxes are not a one-to-one translation across datasets. Without proper correction, the model will learn insufficient adaptations to the anchor boxes and underperform. These models also introduced new combinations of backbones and necks, such as ResNet50 [15], ResNetXt-101 [19] and DarkNet53 [24] as backbones, and FPN [18], PANet [25], Bi-FPN [26] as the necks.

Non-Maximum Suppression

Given that for a single grid cell there are B bounding boxes, if an object exists within the grid cell we must introduce a method of selecting the optimal box. The technique of choice is Non-Maximum Suppression (NMS), which at a high level behaves as a masking operation for bounding boxes above a threshold. Let A, $A \notin B$ denote an optimal predicted bounding box with the highest confidence score, and $\hat{A} \in B$ denote an element of the set of B bounding boxes around the same object. We want to remove all bounding boxes greater than a given threshold, α, in the set B utilizing the Intersection over Union (IoU), Eq. 3.0.1.

$$IoU(A, \hat{A}) = \frac{A \cap \hat{A}}{A \cup \hat{A}} \tag{3.0.1}$$

We first extract A and utilize this as our initial bounding box for comparison. If the IoU of two bounding boxes is greater than α, then it is likely the two objects are representing the same object and we can therefore discard the bounding box in question. We iterate through this process for all bounding boxes in the set B until we are left with a single bounding box for each object in the scene. It is important to consider the potential distribution of objects in the dataset when selecting an IoU threshold as two different objects, that occupy very similar regions of the image space, can be masked by one another if the threshold is too high.

[1] At the time of writing, the YOLOv5 paper is still yet to be published.

3.1.3 Single Shot Detectors

Single Shot Detectors (SSD) [27] strike a balance between R-CNN and YOLO networks as they are generally a simpler network configuration. SSD networks offer improved computational efficiency and speed over R-CNN architectures while providing improved performance over YOLO architectures and more similar inference speeds. The key breakthrough with SSD networks was the discretization of the output prediction space. The network produces a set of default boxes covering different aspect ratios and scales for each discretized location in the feature map. These boxes are similar to anchor boxes utilized in architectures such as YOLOv2 [21] and Faster R-CNN [11], with the key difference being the application across varying resolution feature maps. At inference, the model produces scores for the presence of each object and produces corrections to the final bounding box selection to better encapsulate the identified object [27]. These scores are utilized in a similar fashion to NMS, suppressing the remaining bounding boxes with non-maximum values, or those that do not fit the object at a given aspect ratio. Fig. 3.2 depicts the network configurations of both SSD (top) and YOLO (bottom).

SSD utilizes a VGG-16 [16] backbone, truncating the network prematurely and adding additional layers within the neck of the network. These additional 3×3 convolutional layers produce the set feature maps responsible for the production of default boxes at varying aspect ratios and scales. These feature maps are then sent to the detection head of the network where it can be seen that the number of detections per class scales by a factor of ~ 90 due to the additional predictions across the different resolutions. Predicted bounding boxes are then refined further using NMS over the output space.

3.2 LiDAR Object Detection

LiDAR-based object detection has been at the forefront of automotive technologies as the secondary modality to autonomous systems utilizing cameras. LiDAR sensors produce

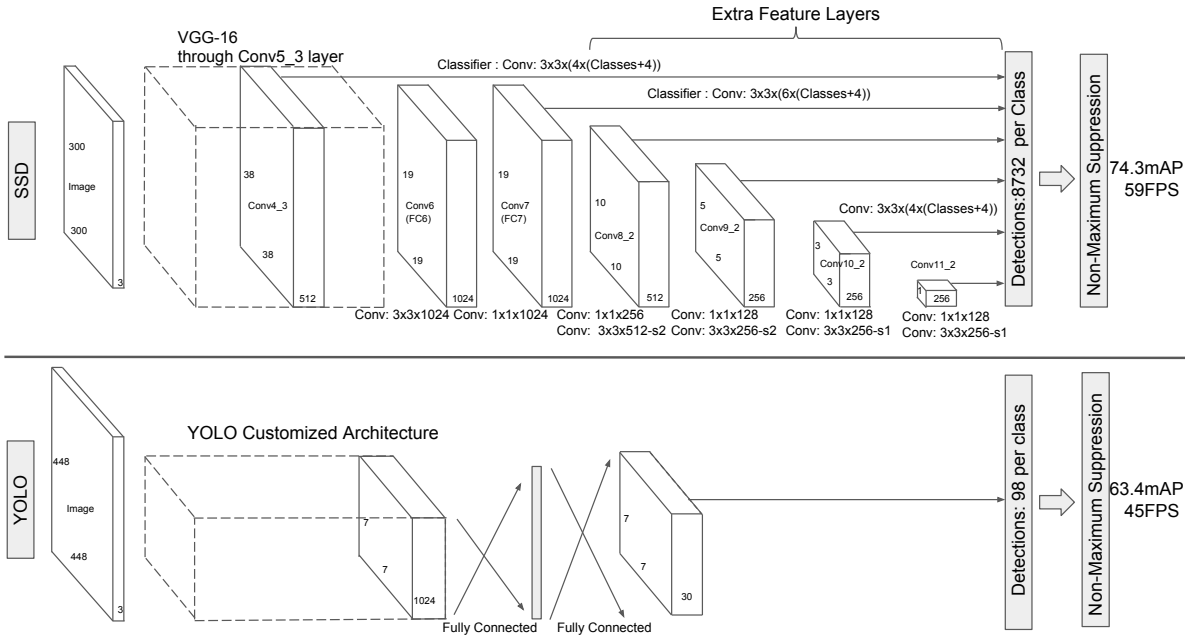


Figure 3.2: **SSD and YOLO Network Comparison:** Comparison of SSD (top) and YOLO (bottom) architectures. The SSD architecture utilizes a VGG-16 [16] backbone and adds additional layers in the neck portion of the network. These additional layers produce the default boxes at various aspect ratios as seen from differing sizes of the convolutional boxes themselves. Note that the YOLO [20] network has since been improved and possesses many differences from the original configuration seen in the figure. Image taken from [27].

360° scans of a scene in which emitted and reflected laser pulses are reconstructed as geometrical points in $\mathcal{R}^3$. This representation can then be translated into various projection schemes, such as a 2D Bird's Eye View (BEV). The most intuitive is the utilization of the raw 3D LiDAR point cloud, applying deep learning networks such as PointNet [28] and its successor PointNet++ [29]. PoinNet++ aims to address the issue of PointNet's inability to capture local structures of objects imposed by the metric space. To overcome this, they introduce a hierarchical structure that applies PointNet recursively on a partitioned point cloud [29]. PointNet treats the PCs as subsets of points, encoding three conditions of subsets in Euclidean space into the network. The first being the fact that points are unordered and therefore the network must retain permutation invariance across the point set. Specifically,

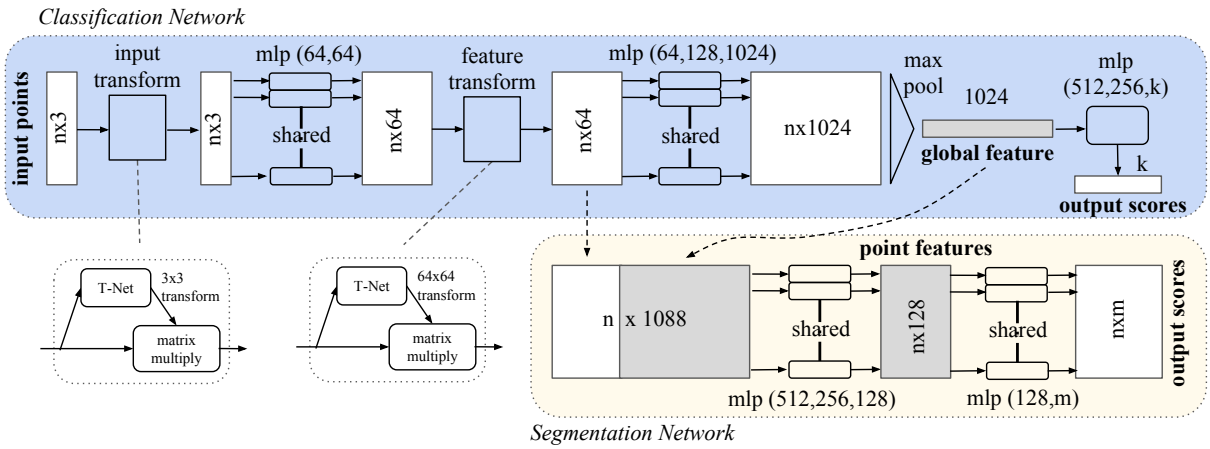


Figure 3.3: **PointNet Architecture:** PointNet utilizes primarily linear layers, treating the input 3D PC as subsets of points. The network consists of two subnetworks, a classification network and a segmentation network, in which much of the design is shared. The classification head transforms, and expands points from a representation in $\mathcal{R}^3$ to a feature representation in $\mathcal{R}^{1024}$ allowing the aggregation of contextual information and geometrical relations between points. This is crucial given that the task of classification of subsets of points relies primarily on their context within the entire PC, not on a point by point basis. These features can then pooled together and processed to produce classification scores. These pooled features are also provided to the segmentation network, which in contrast operates on a point by point basis information wise. Utilizing global context and the relation between points, the PC can then be fully segmented. Image taken from [28].

the ordering of points should not influence the network, only the geometrical locations. Second, the model must not treat inputs as isolated points in space, but rather as localized subsets of connected points given a distance metric defined over the space. Finally, being that we are working with rigid bodies in a metric space, the model must be translationally invariant, or in simpler terms, the rotations of an object, along with any movement through space must not affect the models' ability to learn the geometrical identities of objects. The network uses primarily linear layers to achieve both classification and segmentation of objects in the point cloud and is visualized in Fig. 3.3.

The classification head extracts and transforms features in representations of increasing size, going from a point in $\mathcal{R}^3$ to one in $\mathcal{R}^{1024}$ which are then aggregated using a max-pooling operation. The task of classification of point clouds depends primarily on the context of the points within the global metric space. As such, the aggregation of points via a max pooling

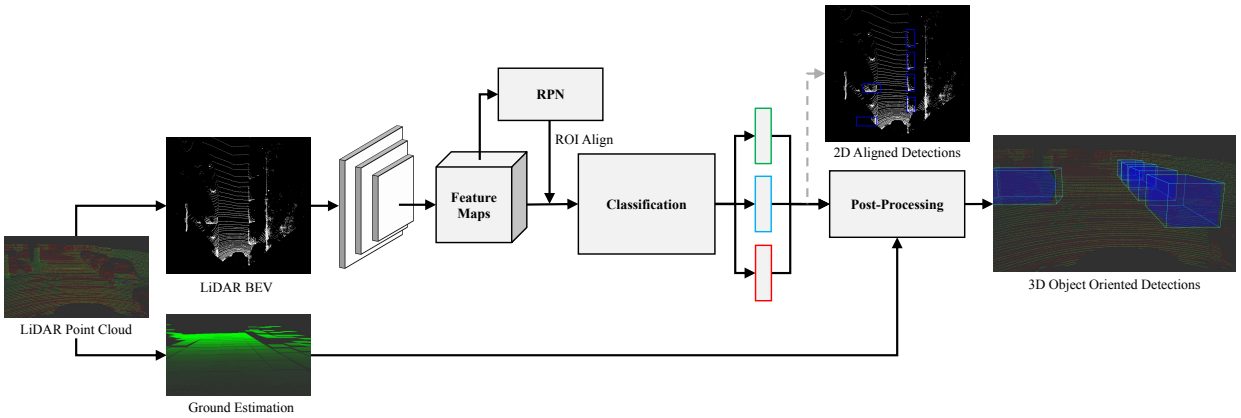


Figure 3.4: **BirdNet Architecture:** The network takes as input a 2D Birds Eye View (BEV) projection of the 3D LiDAR point cloud, processed by a novel cell encoding method. In parallel, estimation of reflections from the ground are constructed in order to remove their effects in the downstream 3D bounding box formation. A Faster R-CNN [11] architecture operates on the BEV image extracting a series of feature maps at a lower resolution, which are then fed to and recombined with outputs from a Region Proposal Network (RPN). This allows efficient classification of only Regions of Interest (ROIs) in the downstream classification head. The resulting output is a 2D projection of bounding boxes into the BEV plane. Utilizing information regarding points estimated in the free driving space, the minimum and maximum z of points contained within the 2D bounding box projection onto the 3D PC, 3D bounding boxes are able to be reconstructed. Image taken from [30].

operation allows efficient classification via a MLP. In order to segment point clouds, one must take into account both the context of the point within the metric space and the local neighborhood of points. This is done via the concatenation of both the aggregated global features from the classification head and the expanded point representations from earlier layers in the classification head. This information can then be combined to extract point-wise features, taking into account the two considerations mentioned prior.

There exists the possibility of deploying different projection schemes to deep learning on LiDAR, such as a top-down projection known as BEV. In BirdNet [30] the authors propose a three-stage pipeline for 3D object detection, utilizing BEV projections as input, and re-projection to the 3D point cloud utilizing outputs from a Faster R-CNN [11] architecture, visualized in Fig. 3.4.

The initial stage of the algorithm focuses on extracting information from the 3D point cloud and encoding it efficiently into a BEV projection via a novel cell encoding pipeline. In

parallel, the algorithm also produces an estimation of reflections coming from the free driving space. When dealing with BEV images, proper classification of 'noise' points can be difficult given the possibility of dense reflections from differing surfaces such as the road, and it is, therefore, important to estimate and remove the effects of these points in downstream object detection tasks. In the second stage, the algorithm then utilizes the 2D BEV image and extracts encoded feature maps. ROIs are extracted from these feature maps and reinjected into the pipeline prior to the classification head, allowing the classification of objects located within the ROIs, and follows the standard functioning of Faster R-CNN. The resulting output is then a series of 2D bounding boxes projected into the BEV image. Each bounding box is given a fixed size as a function of class, derived from statistical analysis. The R-CNN architecture produces class information C, bounding box center (x, y), and a rotation θ, describing the orientation of the object in a coordinate system defined at the box center. The pre-processing step of the algorithm then converts these 2D bounding boxes into 3D projections utilizing information from points contained in the 2D bounding box projection, such as the minimum and maximum z of points within the box. For further information regarding LiDAR object detection we refer the reader to the comprehensive surveys in [31, 32].

3.3 Object Detection utilizing FMCW Radar

The most common practice in recent years is to utilize PC representations [33, 34, 35, 36, 37, 38], inspired by the successes in LiDAR PC object detection. Radar PC representations suffer from pre-processing costs along with loss of information due to filtering, in which implementations of CFAR generally dominate due to its relatively low complexity in comparison to MUltiple SIgnal Classification (MUSIC). Radar configurations contain high levels of noise, which is often hard to discern from object signatures. Radar PC representations are generally sparse due to the need for noise suppression and are therefore not always

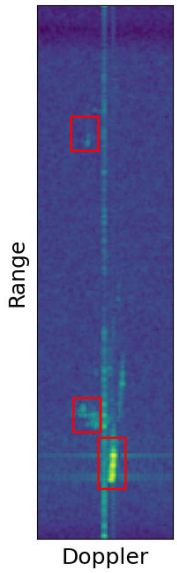


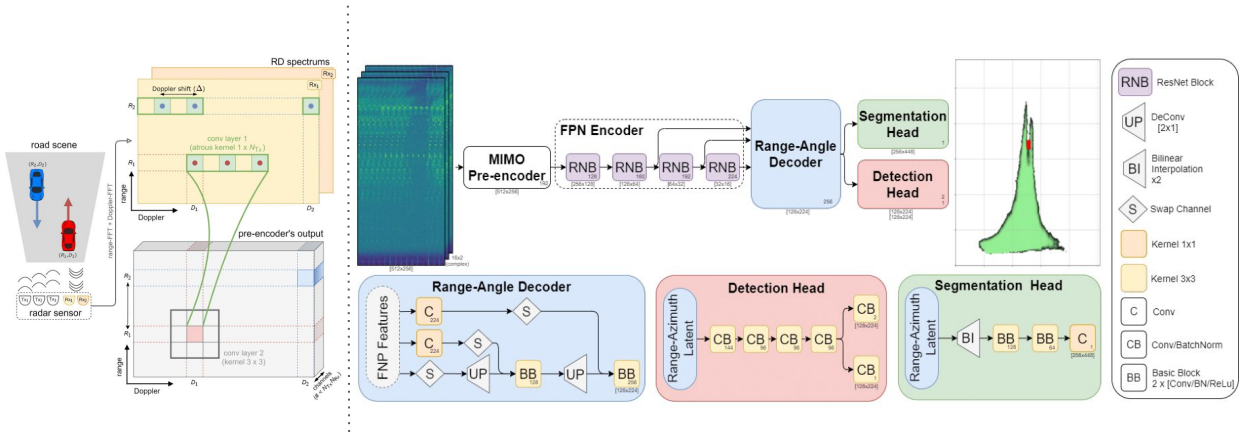
Figure 3.5: **Example Range-Doppler Spectrum:** Dominating frequency components from object reflections appear as 'bright spots' within FMCW radar spectra, information easily picked up by vision based deep learning architectures.

suitable for object detection tasks. As a result, spectrum-based inputs have begun to dominate. The most intuitive choice was the utilization of RAD cubes, a degree of pre-processing prior to the creation of PCs. RAD cubes contain range-Doppler and azimuth signatures of objects and are therefore the most information-dense, allowing efficient object detection [4, 39, 40]. They are also computationally expensive to produce and the resulting networks are more complex given the higher dimensional inputs. A common theme throughout radar architectures is the utilization of image-based object detection networks. RADDet [4] utilizes ResNet like backbone structures coupled to YOLO detection heads, performing object detection and classification with RAD cubes. The utilization of Computer Vision (CV) deep learning architectures proves beneficial given the structure of object signatures in the frequency domain. Objects are represented as 'bright spots' or regions of high density above noise which can easily be interpreted by these models and localized in downstream tasks. Fig. 3.5 depicts an example of object signatures in an RD spectrum.

Raw radar signals in the time domain are high dimensional tensors that possess low interpretability and therefore require some degree of pre-processing for object detection tasks,

usually in the form of FFTs. The utilization of multiple FFTs across Range-Doppler-Azimuth axes allows efficient information extraction in the frequency domain. Transformations across subsequent axes also cause network complexity to scale, *i.e.*, when the full RAD cube is utilized. Thus, the ideal radar network is one which reduces the number of frequency transformations required. Rebut et al.[1] show that networks utilizing RD tensors (FFTRadNet) can approximate this third transformation and efficiently extract angular information. The angular resolution of the radar sensor is directly proportional to the aperture, specifically, the number of T_x and R_x along with their physical spacing. A system with multiple transmitters and receivers is often referred to as a Multiple Input Multiple Output (MIMO) configuration. A clever method of increasing the resolution without physically altering the sensor is to form the *virtual array*, although this comes with a computational cost. FFTRadNet instead utilizes a *MIMO pre-encoder*, a network utilizing Atrous convolutions to aggregate T_x and R_x information in such a way that mimics a virtual array. This allows efficient feature extraction at a higher resolution via a FPN for downstream tasks, such as detection, regression, and free-driving space segmentation. Being that features are extracted from RD spectrums, considerations regarding the axis must be taken into account when features are passed to downstream tasks. FFTRadNet utilizes an upscaling decoder, coupled to the FPN in a U-Net [41] style fashion, in which ordered permutations are inserted to correctly orient the desired representation, *i.e.*, Range-Azimuth. The extracted feature representations are then approximations of object signatures in a polar coordinate system suitable for the network heads to perform classification and regression, Fig. 3.6.

Moreover, they are able to further reduce complexity via the usage of coarse RA grids and regression heads. The center of objects is predicted within a grid $\frac{1}{8}$ and $\frac{1}{4}$ the native azimuth and range resolutions. A secondary regression head then predicts fine-grained correction factors producing localization well within the physical sensor resolutions. Yi et al. [42] further build on this idea, utilizing a Cross-Modal supervision strategy. Manual annotation of radar signals is difficult due to their low interpretability and high noise levels. The authors in



(a) MIMO Pre-encoder (b) FFTRadNet

Figure 3.6: **FFTRadNet Architecture:** FFTRadNet utilizes a MIMO pre-encoder to restructure and realign inputs into a MIMO like spectrum without the computational overhead required by traditional MIMO processing techniques. This is done utilizing atrous convolutions, capable of expanding the FOV of a given convolutional window without the requirement of reformulation and additional parameters. (a) The output of the pre-encoder is processed by a Feature Pyramid Network (FPN) coupled to convolutional based decoder utilizing residual connections across varying resolutions. The output of the decoder is then transformed range-Doppler feature maps in which the axis represent range-Angle information to bed fed to classification and regression heads. (b) Image taken from [1].

[42] develop a novel Fully Convolutional Network (FCN) operating on range-Doppler inputs, which is trained under the supervision of radar-adapted, image segmentation networks. This method alleviates manual annotation and is immune to false targets due to noise. We emphasize that this method uses 'improved' labels when comparisons are presented in later sections. A further study could be the implementation of this training strategy with our Swin based network.

Object detection directly on RD maps has proven to be fruitful, in which the task generally becomes one of segmentation [43, 44, 45, 46]. U-Net [41] and FCN [47] like structures are suitable candidates for such implementation backbones. While this does not provide direct angular information, it allows the extraction of Regions of Interest (ROIs) in the discrete range and Doppler bins. It is then possible to perform the azimuthal FFT over only subsets of the entire signal and extract their angular information. Generally, the input RD maps are

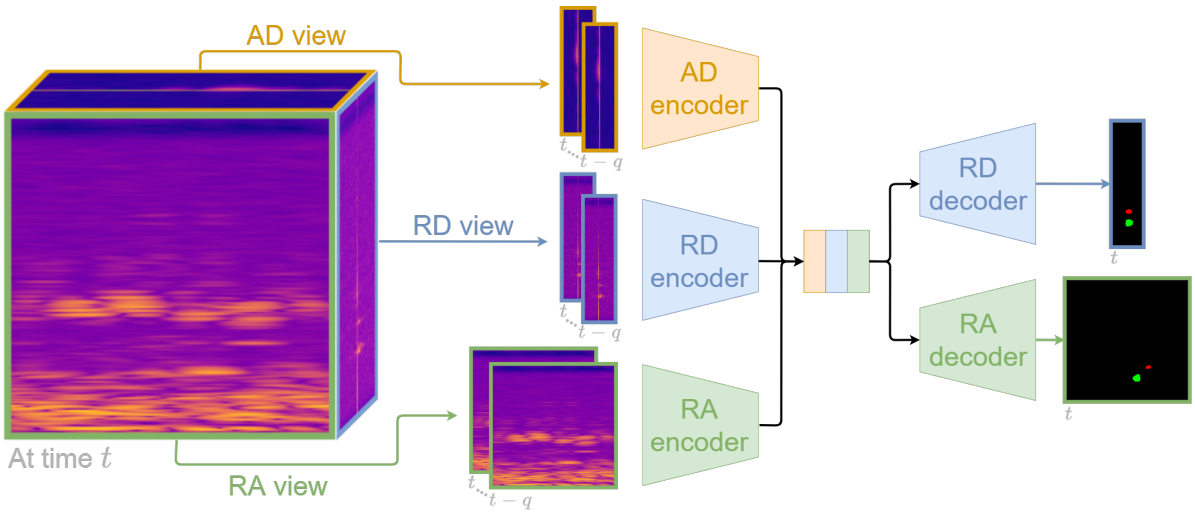


Figure 3.7: **Multi-view Network:** Multi-view Networks utilize two or more RAD cube orientations such as range-Doppler (RD), range-azimuth (RA) and azimuth-doppler (AD). The input is reduced via integration over the reduced axis. The network allows simultaneous object segmentation in both RD and RA orientations. The network utilizes a series of 2D and 3D convolutional layers in a encoder decoder fashion. MV-Net has been improved with various iterations such as the inclusion of Atrous Spatial Pyramid Pooling (ASPP) modules in the bottleneck, allowing increased segmentation across varying resolution feature maps with relatively low complexity increase due to the usage of Atrous convolutions. The model has also been refined to utilize sequenced inputs in the form of channels over a small window of $t - \Delta t$, improving the localization of objects above noise. Image taken from [44].

integrated over the antenna axis to reduce network complexity. It is also possible to combine this information with RA maps [44], known as a Multi-view Network (MV-Net), Fig. 3.7, which are integrated over the chirp, or slow time axis. MV-Net is further improved via the addition of Atrous Spatial Pyramid Pooling (ASPP) modules [48] forming MVA-Net. ASSP modules allow joint feature learning at different resolutions at static network depths without the need for increased kernel sizes or increased complexity [44].

The model takes multiple orientations of the RAD cube, range-Doppler, range-azimuth, and azimuth-doppler, which have been reduced via integration over the removed axis and produces segmentation in both RA and RD spectra. The model was further improved in TMVA-Net, in which the inputs are also time-dependent. The model takes in a short sequence of inputs stacked as channels over some time period $t - \Delta t$, providing temporal

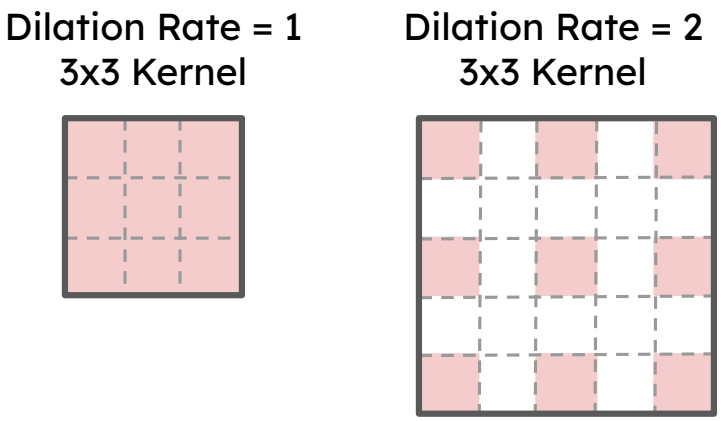


Figure 3.8: **Atrous Convolutions:** Atrous convolutions provide increased kernel FOV without the need for additional parameters. This is done via dilation, in which the specified rate expands the kernel operation to neighbouring regions. The spacing of the expansion grows as a function of the dilation rate.

information of an object's previous locations. The temporal information also allows the network to better learn an object representation over noise. Given that the time period Δt is short, the location and signature of an object will not change drastically and can therefore be thought of as approximately static. However, the high-frequency noise within the channels will be dynamic and ideally more interpretable as noise by the network. The network structure of TMVA-Net is built around 2D and 3D convolutions in an encoder-decoder like structure, in which the encoded features in their respective views are concatenated to form a reduced RAD cube like structure. ASPP modules are inserted in between the encoder and decoder, acting independently on each view before being once again concatenated with the original encodings before being fed to the decoders. The ASPP modules are essentially segmentation networks, providing differing resolution feature maps via the usage of various dilation rates. Dilation essentially broadens the convolutional kernel without the need for redefinement by expanding the FOV of a given kernel. ASPP modules utilize various dilation rates and combine different resolution outputs together via a global average pooling operation and interpolation. A visualization of Atrous convolutions is provided in Fig. 3.8.

Chapter 4

Hierarchical Vision Transformers as Feature Extractors

We build off the architecture proposed in Rebut et al. [1], in which our main contributions are the application of vision transformers to the field of radar object detection, along with the adaption of raw ADC as input. Experiments are performed on radar datasets of differing resolution, in which we discern between the two based on the number of T_x and R_x in the physical configuration. We also perform experiments on different levels of radar pre-processing, namely, the utilization (and lack thereof) of Hamming windowing functions and centering of zero frequency bins along specific axes. Windowing in radar applications allows reduction of spectral leakage in other frequency bins, *i.e.*, reduced side lobe FFT profiles in the frequency domain. In a physical sense, this leakage causes the central object response frequency to be smeared or shifted across discrete bins, altering the physical location interpreted by deep learning networks, but also allows detection of other small sources in differing FFT bins. Centering of frequencies along the Doppler axis specifically is something yet to be explored in its relation to deep learning network performance. In traditional DSP applications, centering of the zero Doppler frequency presents the spectrum from the negative Nyquist frequency to the positive Nyquist frequency, and it avoids displaying a discontinuity in the case of complex signals, where the spectrum is not symmetric between positive and negative frequencies. This is achieved by shifting the positional index of the negative FFT weights. This creates a continuous frequency spectrum across the Doppler

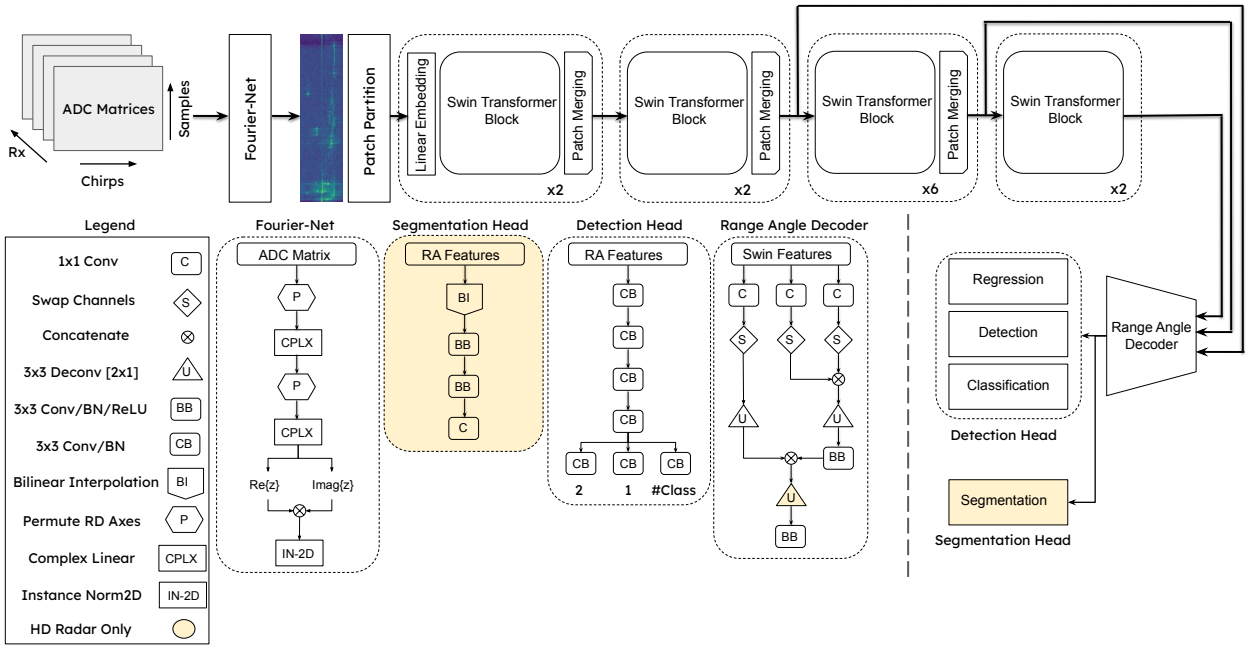


Figure 4.1: **T-FFTRadNet Network Structure:** Overall network structure utilizing raw ADC as input. RD inputs replace the Fourier-Net with range-Doppler FFTs. RAD cube inputs replace the Fourier-Net with range-Doppler-azimuth FFTs and remove channel swapping operations in the Range-Angle Decoder.

frequency response and provides more intuitive responses from echoed objects.

We create T-FFTRadNet, in which the front end of the network in [1] is replaced with a hierarchical Swin Transformer [2]. As mentioned in Sec. 1.3, it is well known that the performance of vision transformers scales with available data. Thus, we deem this research timely as increasingly large radar datasets become available. Moreover, ViTs do not possess spatial inductive bias commonly seen in CNNs and are able to retain a processing pipeline more similar to sequence-to-sequence modeling given the tokenization of patches. The overall network structure is shown in Fig. 4.1.

Utilizing a hierarchical vision transformer (Swin ViT) we are able to extract more prominent features, of varying resolution, to be fed to the network back-end.[1] This is accomplished via the utilization of 2×2 patches, significantly smaller than 3×3 convolutional windows

[1]Hierachichal reffers to the sequential downsampling of inputs.

commonly seen in other FPN, or FCN style networks operating on radar. Moreover, the hierarchical nature of a Swin transformer allows the aggregation of varying resolution features via patch merging between consecutive transformer blocks. Patch merging blocks contain a LayerNorm (LN) operation followed by a simple linear layer and merge information from patches in a 2×2 grid. The 4 patches in the grid are concatenated along the channel axis into a single feature vector of shape $(W/2, H/2, 4C)$, where W and H denote the width and heigth of the combined patches, and C denotes the channel dimensions.[2] This vector is normalized and downsampled via the linear layer producing an output with $2C$ channels, combining the four tokens into two tokens. This results in an increased number of channels relative to the injections shape prior to the Swin block by a factor of two. This action also decreased the height and width of the reshaped output to be injected to the following Swin block. In downstream Swin blocks, Windowed Multi-Head Self-Attention (W-MHSA) allows the network to retain linear complexity with respect to the input. This is due to the fact that self-attention is only computed in local neighborhoods (windows), as opposed to the quadratic complexity required by standard vision transformers when self-attention is computed across all available patches. Sequentially connected layers shift said windows, providing connections between them [2]. Eq. 4.0.1 depicts the flow of outputs between two consecutive blocks, x^l and x^{l+1}, where $\hat{x}$ and x denote the outputs of the W-MHSA, Shifted Window Multi-Head Self-Attention (SW-MHSA), LayerNorm (LN) and MLP modules. This is visually depicted in Fig. 4.2.

$$\begin{aligned}
\hat{\mathbf{x}}^l &= \text{W-MHSA}(\text{LN}(\mathbf{x}^{l-1})) + \mathbf{x}^{l-1} \\
\mathbf{x}^l &= \text{MLP}(\text{LN}(\hat{\mathbf{x}}^l)) + \hat{\mathbf{x}}^l \\
\hat{\mathbf{x}}^{l+1} &= \text{SW-MHSA}(\text{LN}(\mathbf{x}^l)) + \mathbf{x}^l \\
\mathbf{x}^{l+1} &= \text{MLP}(\text{LN}(\hat{\mathbf{x}}^{l+1})) + \hat{\mathbf{x}}^{l+1}
\end{aligned} \tag{4.0.1}$$

[2] Each original patch (token) makes up a quadrant of the grid.

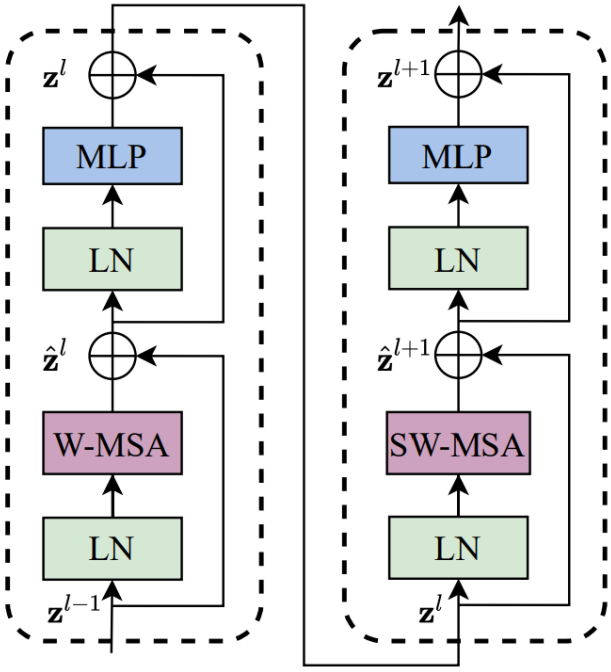


Figure 4.2: **Sequential Swin Transformer blocks:** Visual depiction of two sequential Swin Transformer blocks. In the first block (left), inputs are normalized via a LayerNorm (LN) layer prior to injection to the Windowed Multi-Head Self-Attention (W-MHSA) layer. Windowed attention computes attention in local patches of the input, reducing complexity, and identifying task relevant information. The output of the attention block is then applied to the original feature vector, producing a masked output to be processed via a Multi-Layer Perceptron (MLP). In the second block (right), we repeat a similar process with shifted attention windows, *i.e.*, windows covering portions of data both within and outside of the previous windows. This allows the model to attend to the entire input, extracting relations between all sequences in a more efficient manner. Figure taken from [2].

The Swin blocks utilize W-MHSA (or SW-MHSA) to extract relations between individual tokens, and identify regions of the input in which task relative information resides. This attention mask is then applied to the original input via addition of the two vectors. This operation suppresses low importance regions of the input. The output is then again normalized prior to injection to an MLP, which is tasked with feature extraction from all relevant tokens.

The model is designed to take primarily RD or raw ADC matrices as input, although we show the utilization of RAD cubes in select cases. When raw ADC inputs are utilized, we include transformation layers prior to the Swin transformer. These are discussed in more

detail in Sec. 4.1. Being that transformer style networks utilize functional embeddings, the utilization of varying inputs (such as RD, RAD, etc.) does not drastically increase network complexity.

The Swin transformer extracts varying resolution features which are then sent to the back end of the pre-existing FFTRadNet network. First, the Range-Angle Decoder transforms these features from a compressed RD representation to a decoded RA representation. These decoded features then feed the output heads. The detection head produces a binary grid in the RA coordinate system of the radar sensor such that occupied cells represent the center points of objects. In both HD and LD radar, these grids are reduced representations of the physical sensor limitations to alleviate computational complexity. To compensate for the reduction, the regression head predicts correction factors in both range and angle to allow finer grain resolution for detected objects. We train the binary detection and regression heads utilizing a weighted combination of Focal (detection) and *smooth L1* loss (regression), Eq. 4.0.2, where $\mathbf{y}_b \in [0,1]^{B_{R/n} \times B_{A/m}}$, $\hat{\mathbf{y}}_b \in 0,1^{B_{R/n} \times B_{A/m}}$ and $\mathbf{y}_r \in \mathcal{R}^{B_{R/m} \times B_{A/n}}$, in which m, n denote the range and angular reduction factors. For HD and LD radar, $m = 4, n = 8$ and $m = 4, n = 2$, respectively. α and β are scaling factors set to 2 and 10^2.

$$\begin{aligned} \mathcal{L}_{Bin}(\mathbf{x}, \mathbf{y}_b, \mathbf{y}_r) = \alpha \text{Focal}(\mathbf{y}_b, \hat{\mathbf{y}}_b) + \\ \beta \text{smooth-L1}(\mathbf{y}_r - \hat{\mathbf{y}}_r) \end{aligned} \tag{4.0.2}$$

For the LD radar dataset (RADDet dataset) we add an additional classification head, which produces a likelihood map containing C channels, where C is the number of classes.[3] Given a detection in the binary head, we can then obtain the corresponding classes of detected objects by taking the maximum argument of the likelihood maps at the localized regions subsequent to a *Softmax* function. We train this head independently utilizing pixel-wise cross-entropy loss, Eq. 4.0.3. Where p_k^j is the probability predicted by the model of a given class k, at a specific pixel j. Being that the majority of values in the likelihood masks

[3] In the case of the HD radar dataset, we do not have class labels.

are zeros, the weighting parameter γ must be set to a large value of 10^5 to produce loss contributions on the same orders of magnitude during training.

$$\mathcal{L}_{Class}(\mathbf{x}, \mathbf{y}_c) = -\gamma \sum_{j}^{J} \sum_{k}^{K} y_k^j \log(p_k^j) \tag{4.0.3}$$

For the HD radar dataset (RadIal dataset), we also utilize the segmentation head to predict the free driving space available to the vehicle in a polar coordinate system. As such, the task is defined as a pixel-wise binary classification problem in which binary cross-entropy, Eq. 4.0.4, lends itself naturally. The scale factor λ is set to 10^2.

$$\mathcal{L}_{Free}(\mathbf{y}_{seg}, \hat{\mathbf{y}}_{seg}) = \lambda \sum_{i}^{\frac{B_R}{2}} \sum_{j}^{\frac{B_A}{4}} \mathrm{BCE}\left(\mathbf{y}(i,j), \hat{\mathbf{y}}(i,j)\right) \tag{4.0.4}$$

The total loss contribution then becomes a linear combination of $\mathcal{L}_{Bin}$ and $\mathcal{L}_{Class}$ or $\mathcal{L}_{Free}$.

4.1 ADC Inputs - Linear Transformations

Conventional signal processing on radar relies heavily on the Discrete Fourier Transform (DFT) algorithm, transforming high dimensional FMCW inputs from the time to frequency domain. This transformation also dominates in neural network (NN) based, radar object detection as localizing objects in the time domain is increasingly difficult. DFTs allow a network, specifically a vision-based network, to exploit the dominating frequency components (seen as bright spots) caused by object reflections in downstream tasks. However, utilizing conventional DFTs possess some inherent computational cost, along with the general requirement that some normalization should be performed prior to network injection. An ideal radar object detection network is then one that is capable of utilizing non-normalized inputs in the time domain and learning a transformation suitable to the task. Zhao et al. [3] propose complex-valued linear layers, utilizing the prior knowledge of the Fourier transform. The layers can be seen as an $M \times M$ matrix, where each column acts as a learnable

transformation over a slice of the input $\boldsymbol{x}$ of size $N \leq M$. Thus, mimicking the action of a standard DFT while allowing the network to identify potentially more optimal transformations. Note that permutations are required between consecutive layers to correctly orient the multiplication to mimic a DFT over range or Doppler axis. The weights of the layer are initialized as shown in Eq. 4.0.5, with values corresponding to a standard DFT.

$$w(k,m) = \exp\left(-i\frac{2\pi}{N}km\right), \begin{cases} 0 \leq k \leq M-1 \\ 0 \leq m \leq M-1 \end{cases} \tag{4.0.5}$$

In order to use standard autodiff and backpropogation libraries, the complex valued weights can be decoupled, treating the real and complex components as two real valued weight matrices. For a complex valued vector input $\mathbf{z} = \mathbf{a} + i\mathbf{b}$, we transform through a single linear layer via Eq. 4.0.6.

$$f(W,\mathbf{z}) \mapsto W_{\text{re}}\mathbf{z} + iW_{\text{im}}\mathbf{z} = (W_{\text{re}}\mathbf{a} - W_{\text{im}}\mathbf{b}) + i(W_{\text{im}}\mathbf{a} + W_{\text{re}}\mathbf{b})\,. \tag{4.0.6}$$

It is also preferred that the transformation preserves the continuity of reflected objects in the learned Doppler space, *i.e.*, zero Doppler velocity should be centralized. This is done by shifting the upper and lower half of the weights and is equivalent to the commonly used *fftshift* functions. Initializing the network in such a way provides an initial starting point that has been shown to work well for radar object detection, yet the transformation learned through end-to-end training is likely to no longer be Fourier [3]. In our network, we first inject non-normalized ADC signals to a range transformation layer, followed by a Doppler transformation layer. Given that DFTs in general can produce large values, we utilize a normalization layer before injection into the Swin Transformer. We also experiment with utilizing non-linear activation functions on the outputs of the range and Doppler layers, namely those inspired by Phase-Amplitude (PA) functions which preserve the phase of the activation value. A modified version of *modReLU* [49] is used, in which we utilize

a *LeakyReLU* to preserve negative values of the FFT yet provide non-linearity, Eq. 4.0.7, where z and b are the input vector and bias, respectively.

$$f(z) = \text{LeakyReLU}(|z| + b)\frac{z}{|z|} \tag{4.0.7}$$

In what follows, any experiment utilizing non-linear activation functions on ADC data will be denoted with an NL (Non-Linear) within the respective network title.

Chapter 5

Experimental Results

In what follows, we describe the datasets used within experiments, in which we refer to them as LD and HD radar given their relative numbers of T_x and R_x. We then show the initial design steps of the architecture, motivating the choice of hyperparameters and network component selection. This is followed by a detailed performance analysis on both LD and HD radar.

5.1 Datasets

We utilize a single HD radar dataset, RadIal [1] in which we perform generalized object detection. No class information is produced via the model, only a binary grid (1 being an object, 0 being no object) in a RA coordinate system. The HD radar sensor is composed of 12 transmitters (T_x) and 16 receivers (R_x), in which each T_x produces 256 chirps, each sampled with 512 points. The system is fixed to the front of a vehicle, providing a 180° FOV, a max range of $103m$, and detections of both moving and stationary objects. The range, azimuthal and velocity resolutions are $0.2m$, 0.1° and $0.1\,\mathrm{m\,s^{-1}}$. Objects within the dataset consist of vehicles (cars, trucks, vans, motorcycles, etc.), pedestrians and cyclists. The inputs to the model are either ADC, or RD matrices in which the complex components of the I/Q data are appended as additional channels for the Swin Transformer, producing inputs of shape (512,256,$2R_x$). Note that if the virtual array is formed for the HD radar the

input shape becomes (512,256,$T_x \cdot 2R_x$). Forming this array is computationally expensive given the large number of receivers and transmitters and is therefore excluded in the HD case. This utilization of complex components as additional channels to the Swin Transformer is consistent in all our experiments, except for RAD cubes where the norm of the cubes' values is used, resulting in all real-valued inputs. The training, validation and testing splits are defined as 70%, 15% and 15%, respectively and are provided via a list of indices available in [1].

For LD radar, we utilize RADDet [4]. The sensor is composed of 2 T_x and 4 R_x, where the transmitters produce 64 chirps each sampled 256 times producing inputs of shape (256,64,$T_x \cdot 2R_x$) where a virtual array has been formed. The radar sensor is stationary, capturing both moving and stationary objects in a 180° FOV, with a max range of 50. The range, azimuthal and velocity resolutions are 0.2m, 0.7° and 0.42 m s^{-1}. Objects within the dataset consist of trucks, cars, buses, motorcycles, cyclists and pedestrians. Utilizing the RADDet dataset, we show performance on ADC, RD, and full RAD cube inputs for completeness. In LD radar the model predicts objects and their corresponding classes. In HD and LD datasets that utilize RD or RAD inputs, we normalize the input (I) across respective channels via Eq. 5.0.1, where i, j denote the range-Doppler bin and μ_k, σ_k are the mean and standard deviation of the k^{th} channel. We again note that ADC inputs possess no prior normalization.

$$I_{normalized}(i,j,k) = \frac{I(i,j,k) - \mu_k}{\sigma_k} \tag{5.0.1}$$

We pre-process the raw ADC for the RADDet dataset, producing RD matrices and RAD cubes in the data-loading pipeline via the Intel® MKL numpy interface. RD axes are subject to Hamming windowing functions as shown in Eq. 5.0.2, reducing side lobe profiles. Where N and M represent the number of samples per chirp and the number of chirps, respectively. Experiments are performed with and without windows for the RADDet and RadIal datasets. Note that by default, RadIal has performed windowing.

In both RADDet and RadIal, we perform experiments with and without centering around the zero frequency bin in the Doppler axis (utilizing an FFT shift). When RAD cubes are utilized in RADDet, we also shift the azimuth axis and remove the swapping of channels in the decoder given that inputs are already aligned properly with the desired output axis. Furthermore, we must modify the convolutions slightly to account for the symmetrically shaped output of the Swin transformer. The training and testing split utilized for the LD radar is provided in Zhang et al. [4]. The dataset is split into a 80/20% split for training and testing, in which 10% of the training dataset is reserved for validation. The dataset is split using classwise sampling such that each dataset follows the same distribution of objects.

$$\begin{aligned} W_R &= 0.54 - 0.46\cos\left(2\pi\frac{n}{n-1}\right),\ 0 \leq n \leq N \\ W_D &= 0.54 - 0.46\cos\left(2\pi\frac{m}{m-1}\right),\ 0 \leq m \leq M \end{aligned} \tag{5.0.2}$$

5.2 Initial Model Search

We explore the initial models utilizing Swin ViT components as the feature extraction mechanism. Typical ViT models do not meet the requirements needed to form the feature extractor backbones in our model given that the utilization of an individual token representing all information within the input does not allow the aggregation of varying-resolution feature maps in downstream tasks. Utilization of only a single token may result in the masking of specific objects signatures as their representation exists on a different scale. The work in this section lays the foundation for the finalized model, which we call T-FFTRadNet (Transformer FFTRadNet) and results found in the following sections. All experiments and conclusions drawn are done utilizing the HD radar dataset, with RD inputs, in which we adapt the finalized model LD radar applications to show its generalizability. Note that in plots we commonly refer to T-FFTRadNet in terms of its Swin Encoder component, *i.e.*, Swin Encoder is equivalent to T-FFTRadNet in later sections.

We evaluate the model's performance in comparison to the baseline, FFTRadNet [1].

Table 5.1 and Table 5.2 show comparative results for the Swin-based model (top sub-tables) and FFTRadNet (bottom sub-tables) for the validation and testing datasets. Each model is subject to various IoU thresholds, Eq. 3.0.1, in which we highlight the threshold of 0.5 as the choice used in [1]. An IoU threshold of 0.5 requires the ratio of overlap between A and $\hat{A}$ to be 50% of the set produced via their union for a correct prediction to be counted. It then follows that a lower IoU threshold will produce more 'correct' predictions, although these can be interpreted as less confident, whereas a larger IoU threshold produces higher confidence predictions. This can be seen via the inverse functionality between regressed angle and range errors and IoU threshold. At this stage, we observe performance at all IoU thresholds in order to identify any intrinsic performance characteristics that exist across each model, as opposed to a single IoU of 0.5 as utilized in the literature.

Average Precision (AP), Average Recall (AR) and F1 (harmonic mean between AP and AR), 5.0.3, are the metrics of choice for the task of object detection, in which we take the average recall and precision performance over confidence thresholds. TP,FP and FN denote the true positive, false positive and true negative rates, respectively, where c denotes a specific confidence threshold. Generally these values are taken at face value in the literature, *i.e.*, the standard deviation is not reported. This is due to the fact that the true deviation of these metrics is a function of both the number of confidence thresholds used, and the number of unique objects in each inference. Given that we can select an infinite amount of confidence thresholds on the interval (0,1) we can approximate the deviation to be zero. Instead we are interested in the error of these metrics as function of the number of inference sequences.

$$AP = \sum_{c} \frac{TP(c)}{TP(c) + FP(c)} \quad AR = \sum_{c} \frac{TP(c)}{TP(c) + FN(c)} \quad F1 = 2\frac{AP \cdot AR}{AP + AR} \tag{5.0.3}$$

The number of inference frames within validation and testing sets is relatively small, on

T-FFTRadNet

IoU Threshold	Patch Size	AP (↑)	AR(↑)	F1(↑)	Range Error(m)(↓)	Angle Error(°)(↓)
0.100	(4, 4)	0.963	0.934	0.948	0.177	0.126
0.200	(4, 4)	0.962	0.932	0.947	0.175	0.126
0.300	(4, 4)	0.960	0.931	0.945	0.174	0.125
0.400	(4, 4)	0.950	0.921	0.935	0.167	0.124
0.500	(4, 4)	0.925	0.897	0.911	0.156	0.122
0.600	(4, 4)	0.874	0.848	0.861	0.139	0.118
0.700	(4, 4)	0.757	0.734	0.745	0.112	0.115
0.800	(4, 4)	0.520	0.504	0.512	0.075	0.098
0.900	(4, 4)	0.171	0.166	0.168	0.034	0.060

FFTRadNet

IoU Threshold	AP (↑)	AR(↑)	F1(↑)	Range Error (m)(↓)	Angle Error(°)(↓)
0.100	0.989	0.851	0.915	0.118	0.106
0.200	0.989	0.851	0.915	0.118	0.106
0.300	0.988	0.850	0.914	0.118	0.106
0.400	0.986	0.848	0.912	0.116	0.106
0.500	0.980	0.843	0.906	0.113	0.105
0.600	0.957	0.823	0.885	0.105	0.104
0.700	0.897	0.771	0.829	0.091	0.101
0.800	0.728	0.625	0.673	0.070	0.089
0.900	0.294	0.253	0.272	0.032	0.062

Table 5.1: **Architecture validation data performance:** T-FFTRadNet (top) and FFTRadNet (bottom) validation data results at various IoU thresholds. The highlighted threshold corresponds to the IoU threshold used in Rebut et al.[1].

the order of $\sim 1k$. Each of these frames is also subject to a varying number of objects and we, therefore, quote an upper error bound on the order of 1% for AP, AR, and the F1 score for these datasets via Eq. 5.0.4, where F denotes the above three metrics, and N is the number of samples. We assume a uniform number of objects in each sequence (one object to be specific) in computing this upper bound. In reality, the true error estimates are a function of the object distributions throughout each scene and are much smaller. Therefore, differences on the order of a few σ, *i.e.*, $\sigma - 2\sigma$, can be concluded as significant. We are also interested in both validation and testing sequence performance as this gives us a better idea of true generalizability.

$$\sigma = \sqrt{\frac{F(1-F)}{N}} \tag{5.0.4}$$

In Table 5.1, we see that at an IoU threshold of 0.5 the Swin architecture (T-FFTRadNet) outperforms FFTRadNet in terms of F1 score (the harmonic mean between precision and

T-FFTRadNet

IoU Threshold	Patch Size	AP (↑)	AR(↑)	F1(↑)	Range Error(m)(↓)	Angle Error(°)(↓)
0.100	(4, 4)	0.945	0.904	0.924	0.223	0.134
0.200	(4, 4)	0.939	0.897	0.917	0.215	0.133
0.300	(4, 4)	0.927	0.887	0.907	0.205	0.132
0.400	(4, 4)	0.911	0.871	0.891	0.195	0.129
0.500	(4, 4)	0.864	0.826	0.845	0.173	0.124
0.600	(4, 4)	0.780	0.745	0.762	0.142	0.116
0.700	(4, 4)	0.657	0.628	0.642	0.110	0.107
0.800	(4, 4)	0.458	0.438	0.448	0.074	0.093
0.900	(4, 4)	0.161	0.154	0.157	0.032	0.062

FFTRadNet

IoU Threshold	AP (↑)	AR(↑)	F1(↑)	Range Error(m)(↓)	Angle Error(°)(↓)
0.100	0.985	0.836	0.905	0.127	0.106
0.200	0.985	0.836	0.905	0.127	0.106
0.300	0.985	0.835	0.904	0.127	0.106
0.400	0.978	0.830	0.898	0.123	0.104
0.500	0.968	0.822	0.889	0.118	0.104
0.600	0.940	0.797	0.863	0.108	0.101
0.700	0.872	0.739	0.800	0.092	0.097
0.800	0.700	0.593	0.642	0.068	0.083
0.900	0.310	0.263	0.285	0.032	0.059

Table 5.2: **Architecture testing data performance:** T-FFTRadNet (top) and FFTRadNet (bottom) testing data results at various IoU thresholds. The highlighted threshold corresponds to the IoU threshold used in Rebut et al.[1].

recall) for the validation dataset. Although the regressed range and angle errors are generally higher, they are still within reason given the physical limitations of the radar sensor (0.1° in ϕ, 0.2 m in range). In Table. 5.2 we see that FFTRadNet outperforms at a threshold of 0.5 and above, with regard to F1 score on the testing set, although below 0.5 the two models perform very similarly, albeit FFTRadNet produces range and angle estimates with lower error in general. We also note that FFTRadNet tends to favour precision over recall, *i.e.*, more false negatives to achieve fewer false positives, whereas the Swin architecture seems to be more balanced in this regard.

We visually depict model performance as a function of IoU threshold in Fig. 5.1 in which the columns correspond to the two models, the T-FFTRadNet (left) and FFTRadNet (right). Rows correspond to differing datasets as depicted in the plot titles. We also plot the HD radar sensors' inherent resolutions in both azimuthal angle (ϕ) and range, depicted as dashed horizontal lines in which their colors are the same as the error produced by each model (solid lines). Each plot also contains the free-driving-spaces scores, mean IoU (mIoU), in which

description	symbol	value
Classification Loss Weight	α	2
Regression Loss Weight	β	100
Free-space Loss Weight	λ	100
Learning Rate	δ	10^{-4}
Learning Rate Decay Steps	S	10
Learning Rate Decay Rate	ϵ	0.9
Embedding Dimension	C	48
Encoder Block Depth	$[D_1, D_2, D_3, D_4]$	$[2, 2, 6, 2]$
Number of Attention Heads	$[H_1, H_2, H_3, H_4]$	$[3, 6, 12, 24]$
Window Size	W	7
Patch Size	P	4×4

Table 5.3: **Baseline hyperparameters of the Swin architecture:** these values are not fine-tuned, but have shown to be reliable as an initial starting point.

the output is a binary regression task subject to a threshold of 0.5. The mean of all outputs within the dataset is utilized as the final metric.

As mentioned prior, the gap that exists between AP and AR for the FFTRadNet model shows favoritism towards the acceptance of false negatives over false positives, in comparison to the Swin architecture in which this gap exists on a smaller scale showing that the two metrics are more equally balanced. The Swin model significantly outperforms FFTRadNet in terms of the free-space-driving metric (mIoU), in which we see a $5-10\%$ increase depending on the dataset. We have utilized the same loss weighting scheme as discussed in Rebut et al. [1], and explore a different loss weighting scheme in the section to follow as these directly impact the balance between object detection and free-space-driving performance. Table 5.3 details the current architecture hyperparameters.

5.2.1 Loss Weighting

We adjust the loss weighting parameters to see the performance tradeoff between free-driving-space and object detection metrics (AP,AR,$F1$). The updated network parameters can be found in Table 5.6, in which we decrease the regression and free-space weights by an order of magnitude in comparison to the results previous. All other hyperparameters in the

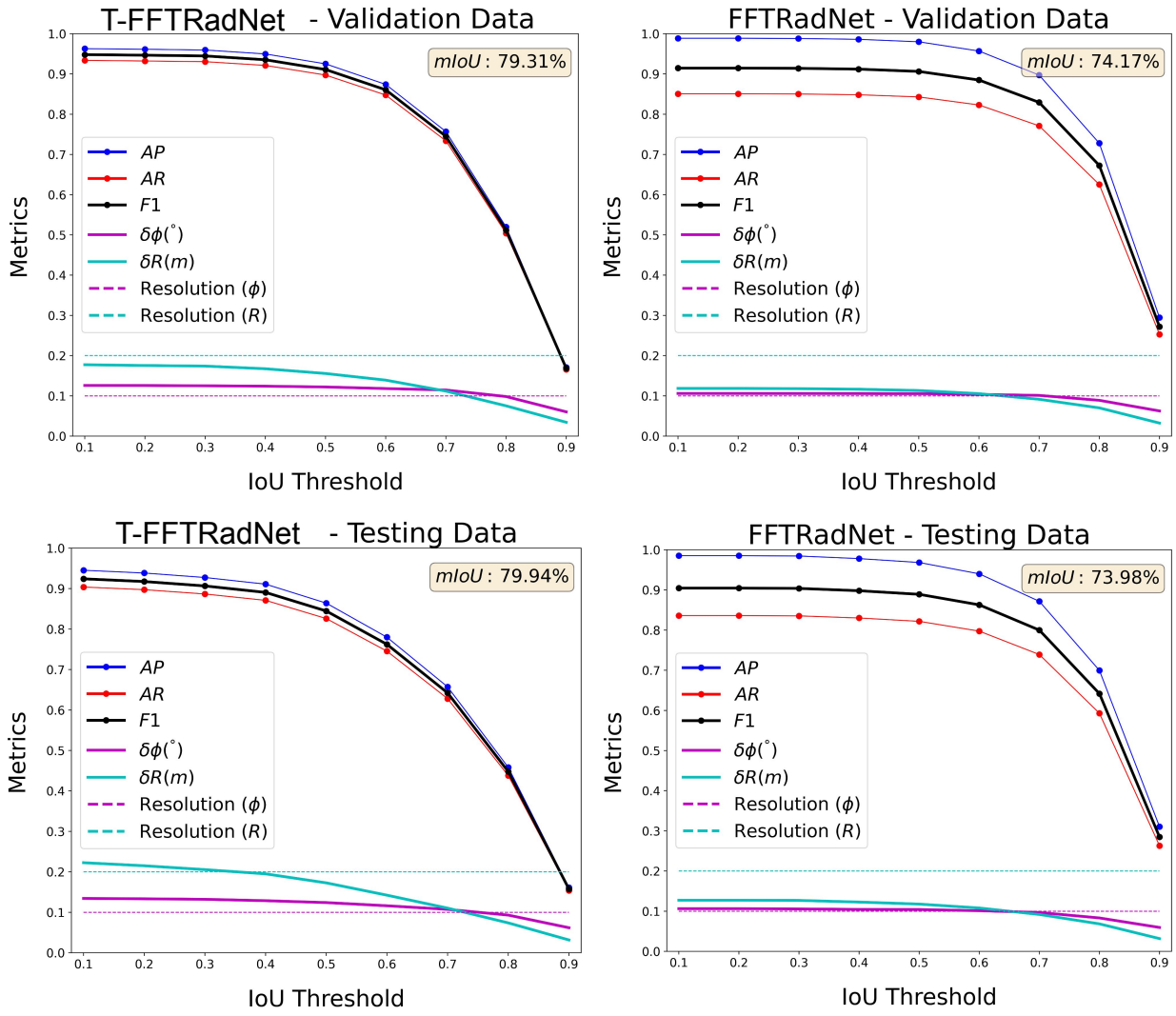


Figure 5.1: **Evaluation metrics as a function of IoU threshold:** Plots of Average Precision (AP), Average Recall (AR), F1 Score, Angle Error ($\delta\phi$), and Range Error (δR). The azimuthal and angular resolution of the HD radar sensor are overlayed with coordinated color schemes. Rows correspond to datasets at inference, validation (top) and testing (bottom). Columns correspond to models, Swin ViT [2] (left) and FFTRadNet [1] (right). Angular and range errors are inversely proportional to the IoU threshold, which is expected given that more potential candidates will be proposed as bounding boxes. mAP, mAR and F1 all are also inversely proportional as we are now including more predictions in our final output. Notice the T-FFTRadNet model has a more explicit dependence on threshold, seen by the pronounced curve. However, T-FFTRadNet also greatly outperforms in terms of free driving space estimates, denoted by the mIoU metric in the included in each plot.

Reweighted Loss Contributions

IoU Threshold	Patch Size	AP (↑)	AR(↑)	F1(↑)	Range Error(m)(↓)	Angle Error(°)(↓)
0.100	(4, 4)	0.979	0.934	0.956	0.184	0.137
0.200	(4, 4)	0.979	0.933	0.955	0.183	0.137
0.300	(4, 4)	0.974	0.928	0.950	0.179	0.135
0.400	(4, 4)	0.961	0.916	0.938	0.171	0.134
0.500	(4, 4)	0.933	0.889	0.910	0.157	0.133
0.600	(4, 4)	0.869	0.829	0.849	0.136	0.129
0.700	(4, 4)	0.754	0.718	0.736	0.110	0.119
0.800	(4, 4)	0.510	0.486	0.497	0.072	0.100
0.900	(4, 4)	0.182	0.174	0.178	0.033	0.065

Default Loss Weighting

IoU Threshold	Patch Size	AP (↑)	AR(↑)	F1(↑)	Range Error(m)(↓)	Angle Error(°)(↓)
0.100	(4, 4)	0.963	0.934	0.948	0.177	0.126
0.200	(4, 4)	0.962	0.932	0.947	0.175	0.126
0.300	(4, 4)	0.960	0.931	0.945	0.174	0.125
0.400	(4, 4)	0.950	0.921	0.935	0.167	0.124
0.500	(4, 4)	0.925	0.897	0.911	0.156	0.122
0.600	(4, 4)	0.874	0.848	0.861	0.139	0.118
0.700	(4, 4)	0.757	0.734	0.745	0.112	0.115
0.800	(4, 4)	0.520	0.504	0.512	0.075	0.098
0.900	(4, 4)	0.171	0.166	0.168	0.034	0.060

Table 5.4: **Swin architecture validation data performance comparison:** Reweighted loss contributions (top), default weighting scheme (bottom) comparison. The highest F1 score at an IoU threshold of 0.5 is highlighted in cyan.

network remain identical. Table 5.4 and Table 5.5 contain the resulting performance metrics from this study for the validation, and testing datasets, respectively. The top sub-table corresponds to the updated loss weights, the bottom sub-table corresponds to the results from the default weighting from the previous section.

We further compare the two models in a similar fashion to the previous section via visualization of the above metrics as a function of IoU threshold in Fig. 5.2. Rows correspond to differing datasets, namely, validation (top) and testing (bottom). The left column corresponds to the Swin model with default loss weighting (see Table 5.3), the right columns correspond to the updated weighting. We again plot the HD radar sensors' inherent resolutions in both azimuthal angle (ϕ) and range, depicted as dashed horizontal lines in which their colors are the same as the error produced by each model (solid lines). Each plot will also contain the free-driving-spaces scores, mean IoU (mIoU), in which the output is a binary regression task subject to a threshold of 0.5.

Considering the results in Tables 5.4 and 5.5, along with the visualizations in the Fig. 5.2,

Reweighted Loss Contributions

IoU Threshold	Patch Size	AP (↑)	AR(↑)	F1(↑)	Range Error(m)(↓)	Angle Error(°)(↓)
0.100	(4, 4)	0.967	0.903	0.934	0.226	0.148
0.200	(4, 4)	0.963	0.899	0.930	0.221	0.147
0.300	(4, 4)	0.951	0.887	0.918	0.212	0.144
0.400	(4, 4)	0.925	0.863	0.893	0.196	0.142
0.500	(4, 4)	0.881	0.823	0.851	0.175	0.138
0.600	(4, 4)	0.795	0.742	0.768	0.145	0.132
0.700	(4, 4)	0.660	0.616	0.637	0.113	0.120
0.800	(4, 4)	0.429	0.401	0.415	0.073	0.099
0.900	(4, 4)	0.159	0.148	0.153	0.033	0.066

Default Loss Weighting

IoU Threshold	Patch Size	AP (↑)	AR(↑)	F1(↑)	Range Error(m)(↓)	Angle Error(°)(↓)
0.100	(4, 4)	0.945	0.904	0.924	0.223	0.134
0.200	(4, 4)	0.939	0.897	0.917	0.215	0.133
0.300	(4, 4)	0.927	0.887	0.907	0.205	0.132
0.400	(4, 4)	0.911	0.871	0.891	0.195	0.129
0.500	(4, 4)	0.864	0.826	0.845	0.173	0.124
0.600	(4, 4)	0.780	0.745	0.762	0.142	0.116
0.700	(4, 4)	0.657	0.628	0.642	0.110	0.107
0.800	(4, 4)	0.458	0.438	0.448	0.074	0.093
0.900	(4, 4)	0.161	0.154	0.157	0.032	0.062

Table 5.5: **Swin architecture testing data performance comparison:** Reweighted loss contributions (top), default weighting scheme (bottom) comparison. The highest F1 score at an IoU threshold of 0.5 is highlighted in cyan.

description	**symbol**	**value**
Classification Loss Weight	α	2
Regression Loss Weight	β	10
Free-space Loss Weight	λ	10
Learning Rate	δ	10^{-4}
Learning Rate Decay Steps	S	10
Learning Rate Decay Rate	ϵ	0.9
Embedding Dimension	C	48
Encoder Block Depth	$[D_1, D_2, D_3, D_4]$	$[2, 2, 6, 2]$
Number of Attention Heads	$[H_1, H_2, H_3, H_4]$	$[3, 6, 12, 24]$
Window Size	W	7
Patch Size	P	4×4

Table 5.6: **Updated hyperparameters of the Swin architecture:** classification, regression and free-space loss weights are adjusted to see functionality in relation to different metrics. This is done by decreasing the regression and free-space-driving loss weights by and order of magnitude ($100 \rightarrow 10$). All other hyperparameters are left identical.

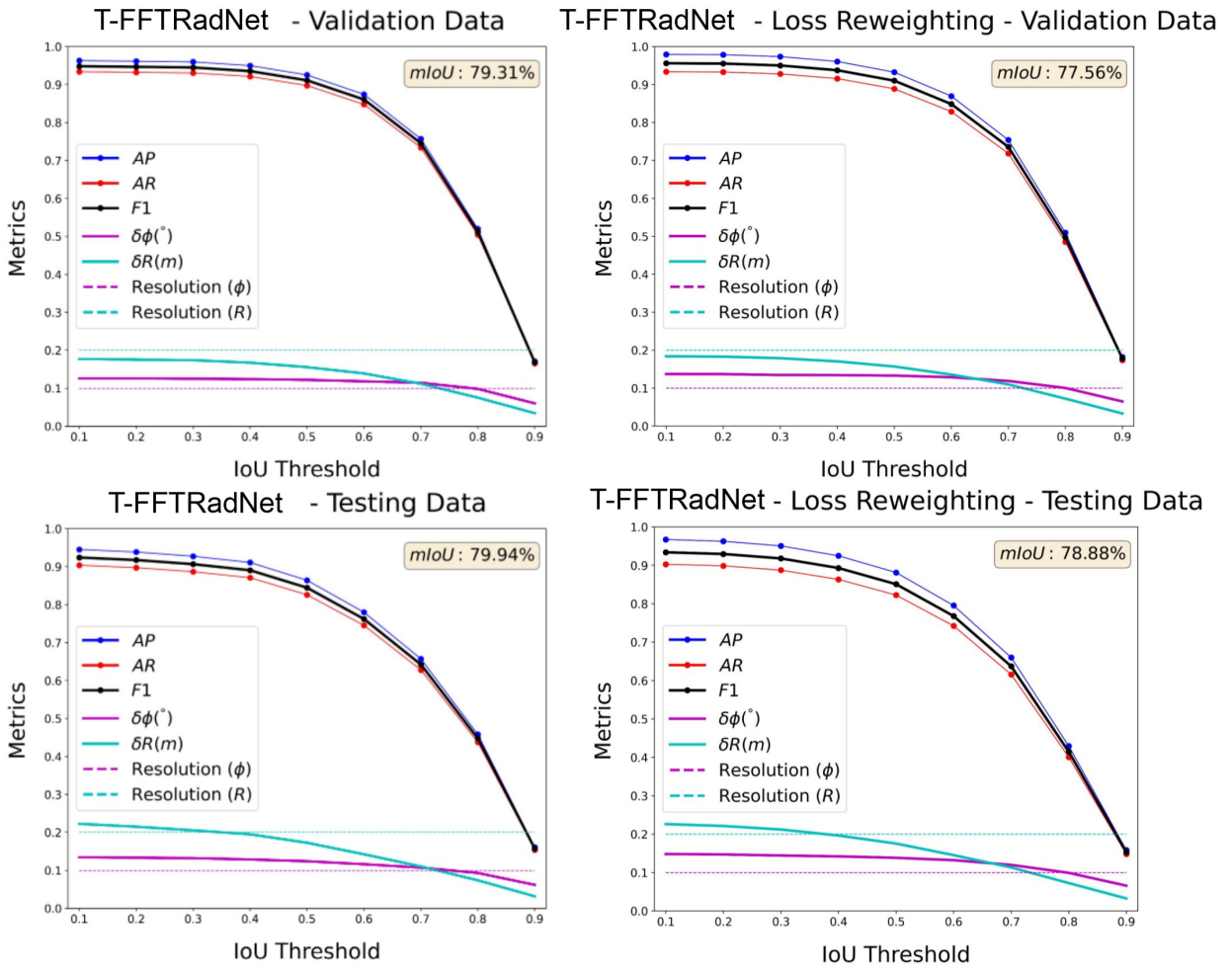


Figure 5.2: **Evaluation metrics as a function of IoU threshold, T-FFTRadNet reweighting comparison:** Plots of Average Precision (AP), Average Recall (AR), F1 Score, Angle Error ($\delta\phi$), and Range Error (δR). The azimuthal and angular resolution of the HD radar sensor are overlayed with coordinated color schemes. Rows correspond to datasets at inference, validation (top) and testing (bottom). Columns correspond to models with different loss weightings, original loss weighting (left) and adjusted (right). Angular and range errors are inversely proportional to the IoU threshold, which is expected given that more potential candidates will be proposed as bounding boxes. AP, AR and F1 all are also inversely proportional as we are now including more predictions in our final output.

we see there exists no significant increase in classification performance regarding F1. The mIoU metric also tends to decrease as a result of the adjusted weights on the order of σ. As a result, the default loss weighting scheme is deemed sufficient and further improvement needs to be obtained via architecture adjustments. Note that in this study we opted not to explore increasing the weighting parameter given that doing so causes the two loss contributions to inhibit convergence of the network given their order of magnitude being significantly larger throughout all points of training.

5.2.2 Doubling Embedding Dimension

A key hyperparameter in Swin ViTs is the embedding dimension, effectively controlling the dimension of the feature map each patch is encoded into. Due to the non-linear complexity of transformer embedding schemes, the parameter count of the model does not scale linearly with the input embedding, *i.e.*, an embedding dimension of 96 does not correspond to a model twice as large as a model with an embedding dimension of 48. Ideally, we want to minimize the number such that we allow sufficient information extraction while remaining on the same order of magnitude parameter-wise as FFTRadNet. A value of 48 allows the model to retain complexity similar to that of FFTRadNet in terms of parameter count. In what follows we explore doubling the embedding dimension from 48 to 96 and explore the model performance as a result. Table. 5.7 provides a numerical comparison of the two models with differing embedding dimensions. Note that in this table we show only results utilizing an IoU threshold of 0.5 as other threshold results show a similar pattern and are redundant at this stage. The utilization of only an IoU threshold of 0.5 will be retained throughout all sections from here on. Fig. 5.3 depicts the results visually across all thresholds.

Dataset	Embedding Dimension (C)	AP (↑)	AR(↑)	F1(↑)	Range Error(m)(↓)	Angle Error(°)(↓)	mIoU (↑)
Validation	96	0.938	0.886	0.911	0.151m	0.126°	79.81%
Validation	48	0.925	0.897	0.911	0.156m	0.122°	79.31%
Testing	96	0.880	0.828	0.853	0.156m	0.146°	80.44%
Testing	48	0.864	0.826	0.845	0.173m	0.124°	79.94%

Table 5.7: **Embedding Dimension Comparison:** Increasing embedding dimension with a patch size of 4×4 does slightly increase overall performance, although promotes excessive network parameter growth. Given the magnitude of performance increase, we instead opt for a smaller embedding dimension to keep network scale on the same order of magnitude as FFTRadNet[1].

From the results in the table, the relative improvement is marginal and does not warrant the increased computational complexity introduced via doubling the embedding layer. Doubling the embedding dimension increases the parameter count of the network from $\sim 9 \times 10^6$ to $\sim 30 \times 10^6$, a significant increase for marginal performance.

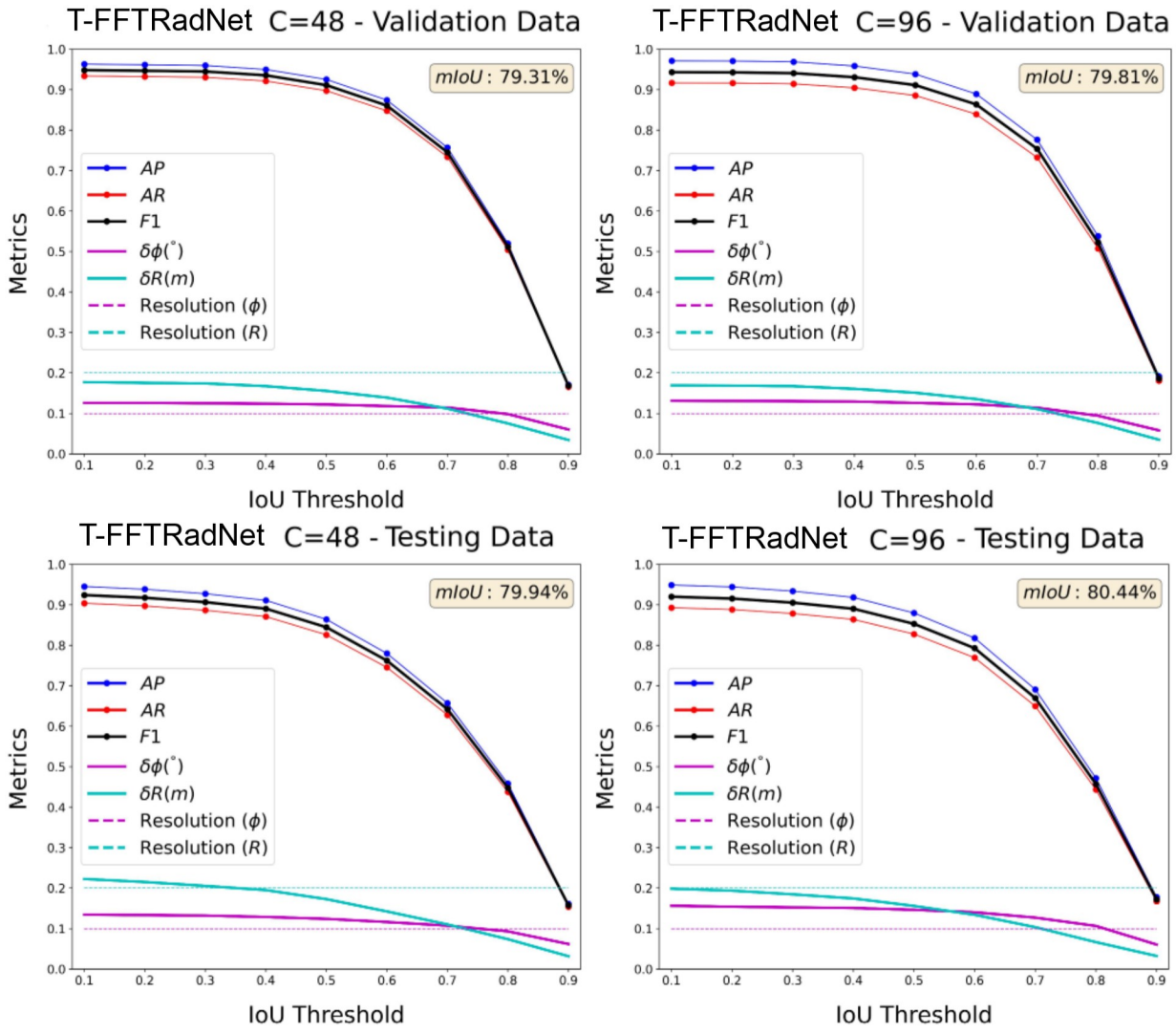


Figure 5.3: **Evaluation metrics as a function of IoU threshold, Swin embedding dimension comparison:** Plots of Average Precision (AP), Average Recall (AR), F1 Score, Angle Error ($\delta\phi$), and Range Error (δR). The azimuthal and angular resolution of the HD radar sensor are overlayed with coordinated color schemes. Rows correspond to datasets at inference, validation (top) and testing (bottom). Columns correspond to Swin models with different embedding dimensions (C), original Swin ViT (C = 48,left) and adjusted (C = 96,right). Angular and range errors are inversely proportional to the IoU threshold, which is expected given that more potential candidates will be accepted as bounding boxes. mAP, mAR and F1 all are also inversely proportional as we are now including more predictions in our final output.

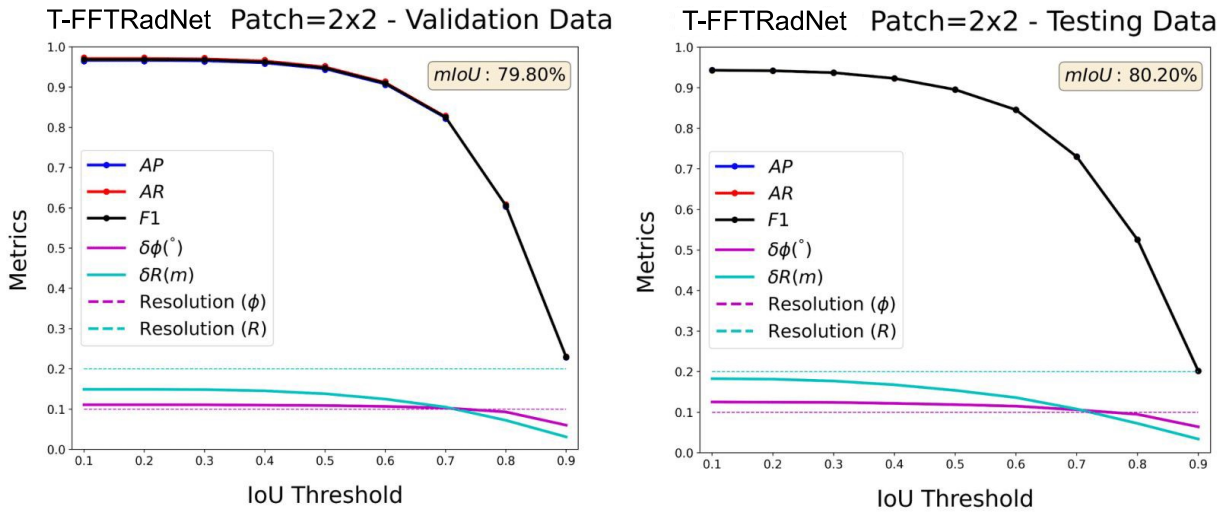


Figure 5.4: **Evaluation metrics as a function of IoU threshold, reduced patch size:** Swin architecture with reduced patch size of 2x2 for the validation (left) and testing (right) datasets. Note the equivalence of the two metrics, mAP and mAR, resulting in an F1 score approximately equal to either.

5.2.3 Decreased Patch Size

In order to further improve performance we investigate the utilization of decreased patch sizes. Given that Swin ViTs are designed to operate on images, the input grid is generally larger. In image data, a patch size of 4×4 corresponds to small local neighborhoods of pixels to be combined into a single patch. In the case of radar, inputs grids (chirp and samples axis) are relatively smaller, and therefore a smaller patch size may be more efficient given information extraction. Moreover, a smaller patch size operating in the frequency domain helps resolve objects whose peaks tend to overlap, *i.e.*, two objects under the same peak where one object forms a shouldered bump off the main frequency peak. Increasing the patch size would only inhibit the models ability to localize differing frequency profiles who are closely spaced. Therefore, we opt to instead explore a decreased patch size. Fig. 5.4 depicts the results visually and Table 5.12 contains numerical values at a threshold of 0.5.

We see significant performance increases when a 2×2 patch is used.[1] Interestingly, we see a large increase in AP and AR at a threshold of 0.5 coupled to a relatively less

[1]Note that a patch smaller than this such as 1×1 defeats the purpose of utilizing computer vision style networks.

Dataset	Patch Size	AP (↑)	AR(↑)	F1(↑)	Range Error(m)(↓)	Angle Error(°)(↓)	mIoU(↑)
Validation	2×2	0.943	0.952	0.948	$0.14m$	0.11°	79.80%
Validation	4×4	0.925	0.897	0.911	$0.16m$	0.12°	79.31%
Testing	2×2	0.896	0.895	0.895	$0.15m$	0.12°	80.20%
Testing	4×4	0.864	0.826	0.845	$0.17m$	0.12°	79.94%

Table 5.8: **Reduced patch size results:** Results utilizing a reduced patch size of 2×2.

description	**symbol**	**value**
Focal Loss Weight	α	2
Regression Loss Weight	β	10^2
Free-space Loss Weight	λ	10^2
Pixel-wise BCE Weight	γ	10^5
Learning Rate	δ	10^{-4}
Learning Rate Decay Steps	S	10
Learning Rate Decay Rate	ϵ	0.9
Embedding Dimension	C	48
Encoder Block Depth	$[D_1, D_2, D_3, D_4]$	$[2, 2, 6, 2]$
Number of Attention Heads	$[H_1, H_2, H_3, H_4]$	$[3, 6, 12, 24]$
Window Size	W	7
Patch Size	P	2×2

Table 5.9: **T-FFTRadNet finalized hyperparameters:** The set of semi-optimal hyperparameters identified for use in experiments to follow. We have included the classification loss weight (Pixel-wise BCE) for completeness but note this only pertains to LD radar where labels exist.

significant increase in free-driving-space segmentation. Also note the equivalence of the two metrics, AP and AR, throughout all IoU thresholds in Fig. 5.4. This equivalence is not observed in FFTRadNet. A decreased patch size also removes any need for increased embedding dimensions, as doing so only prolongs model convergence to a spot providing identical performance. We define a set of semi-optimal hyperparameters in Table 5.9 that are used throughout the remainder of experiments.

5.2.4 Extras: Swin Decoder Blocks

The Swin architecture in the prior section proves to work well as a feature extractor feeding the Range Angle Decoder used to approximate the Range Angle information. We include the following information as extras, relating to other avenues explored in the model refinement process. Namely, different configurations of the Range-Angle Decoder attempting to utilize Swin transformer blocks in replacement of existing convolution blocks for the combining of information from skip connections. Fig. 5.5 depicts the convolution-based decoder structure (left), and the modified structure utilizing Swin blocks (right). Operations are colour-coordinated and defined in the legend (center). The number of Swin layers in each block, along with the number of attention heads used are defined in the figure adjacent to the corresponding block.

We perform the same analysis as in previous sections, evaluating different object detection metrics as a function of IoU threshold. The results from the three different datasets, validation (top row) and testing (bottom row), are tabulated in Table 5.10. Fig. 5.6 provides visualization of the same metrics as a function of IoU threshold.

From inspection of Table 5.10 and Fig. 5.6 it is apparent that the convolutional decoder from previous sections (see Table 5.12 and Fig. 5.4) provides better results. The main reason for this is likely the need to perform axis swapping as the axis representation sent to the detections heads should be range-azimuth-Doppler, as opposed to the decoder input which is range-Doppler-azimuth. To utilize Swin blocks we must perform additional axis transposition in order to properly shape inputs. It is likely these additional operations incur a loss of generalized range-azimuth object detection information. Moreover, the network becomes over-parameterized and is prone to overfitting, evident via diverging validation loss. The decoder is not able to learn generalized features capable of being utilized at inference. A decoder architecture utilizing strictly convolutions is much more suited to the transformations needed to approximate the azimuthal components. This is also true in terms of free-space scores, in which we see a degradation in performance utilizing the Swin blocks,

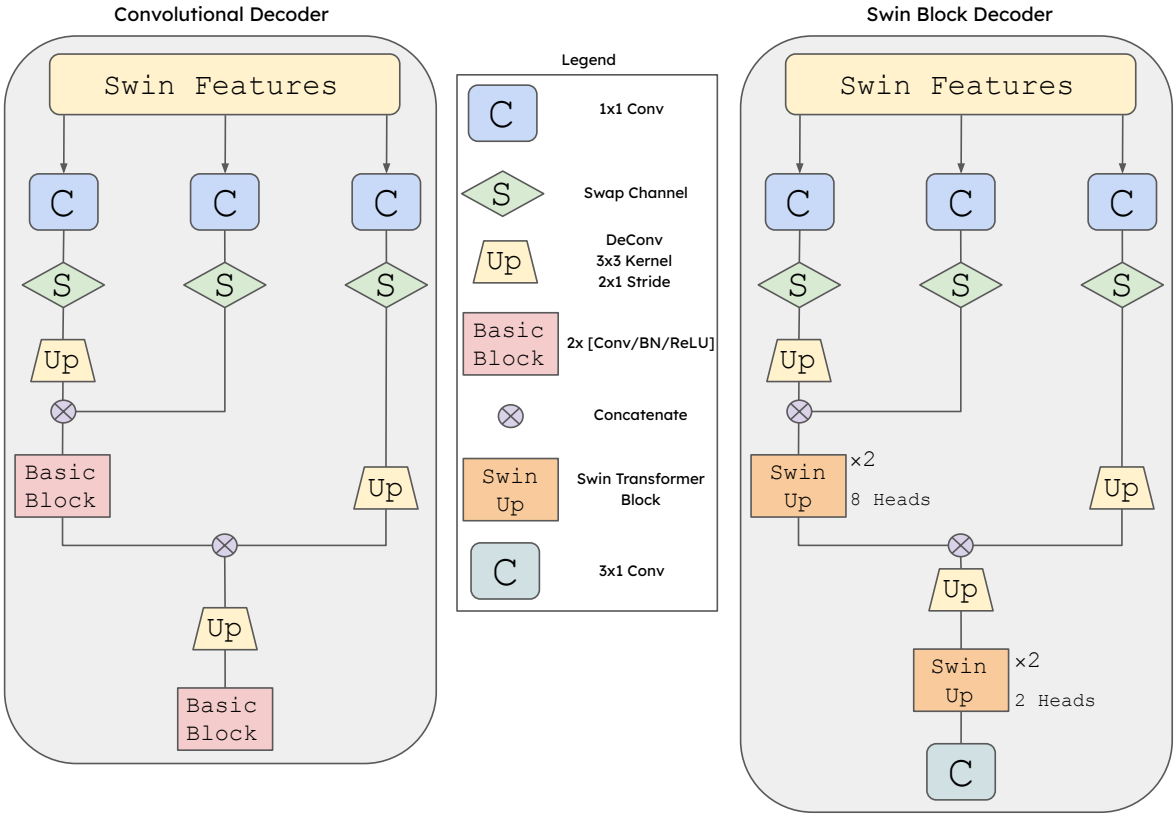


Figure 5.5: **Range Angle Decoder Structures:** Convolutional decoder (left), Swin block decoder (right). Components are colour-coordinated and defined in the legend (center). The number of transformer layers in each block, along with the number of attention heads in the Swin blocks is defined adjacent to the corresponding block.

although the degree is much less severe. The lesser degradation seen in the free-space scores is due to this task being generally easier. The model can arbitrarily predict large regions of free or occupied pixels and incurs a decent performance by chance.

Swin Decoder Blocks - Validation Data

IoU Threshold	Patch Size	AP (↑)	AR(↑)	F1(↑)	Range Error(m)(↓)	Angle Error(°)(↓)
0.100	(4, 4)	0.499	0.395	0.441	0.589	0.580
0.200	(4, 4)	0.425	0.338	0.376	0.484	0.548
0.300	(4, 4)	0.340	0.270	0.301	0.383	0.489
0.400	(4, 4)	0.259	0.205	0.229	0.306	0.422
0.500	(4, 4)	0.177	0.140	0.156	0.228	0.351
0.600	(4, 4)	0.118	0.093	0.104	0.169	0.295
0.700	(4, 4)	0.059	0.046	0.052	0.112	0.205
0.800	(4, 4)	0.027	0.021	0.024	0.073	0.132
0.900	(4, 4)	0.007	0.005	0.006	0.040	0.072

Swin Decoder Blocks - Testing Data

IoU Threshold	Patch Size	AP (↑)	AR(↑)	F1(↑)	Range Error(m)(↓)	Angle Error(°)(↓)
0.100	(4, 4)	0.456	0.351	0.397	0.560	0.674
0.200	(4, 4)	0.380	0.294	0.331	0.439	0.611
0.300	(4, 4)	0.314	0.241	0.273	0.356	0.547
0.400	(4, 4)	0.238	0.182	0.206	0.270	0.490
0.500	(4, 4)	0.172	0.130	0.148	0.211	0.412
0.600	(4, 4)	0.110	0.083	0.095	0.154	0.334
0.700	(4, 4)	0.061	0.047	0.053	0.107	0.257
0.800	(4, 4)	0.022	0.017	0.019	0.064	0.159
0.900	(4, 4)	0.004	0.004	0.004	0.021	0.086

Table 5.10: **Swin decoder block results:** Object detection metrics for validation (middle) and testing (bottom) datasets at different IoU thresholds utilizing Swin Decoder blocks in the Range Angle Decoder. Transformer blocks are merged into the convolution-based framework, replacing convolutional blocks used to combine information from skip connections. Input patches are expanded in an inverse fashion to the patch merging layer used in the encoder blocks.

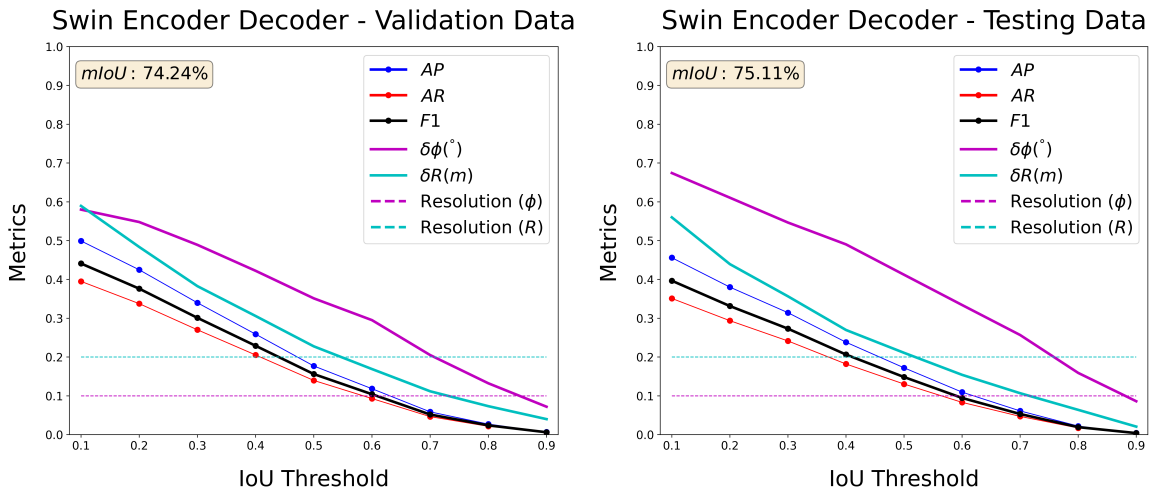


Figure 5.6: **Evaluation metrics as a function of IoU threshold, Swin decoder blocks:** Plots of Average Precision (AP), Average Recall (AR), F1 Score, Angle Error ($\delta\phi$), and Range Error (δR). The azimuthal and angular resolution of the HD radar sensor are overlayed with coordinated color schemes. Columns correspond to datasets at inference, validation (left) and testing (right). We see the utilization of transformer blocks in the decoder severely degrades overall performance. The model is likely over-parameterized and would need sufficient regularization, although the increased complexity makes this architecture inferior to the convolutional-based decoder.

Model	Dataset	FLOPs ↓	# Params. ↓
FFTRadNet [1]	RadIal	288 G	3.79 M
Cross Modal DNN [42]	RadIal	358 G	7.7 M
T-FFTRadNet	RadIal	194 G	9.64 M
T-FFTRadNet	RADDet	10 G	7.5 M
T-FFTRadNet (RAD)	RADDet	26 G	8.22 M
RADDet (RAD) [4]	RADDet	25 G	9.59 M

Fourier Transforms

Model	Dataset	FLOPs ↓	# Params. ↓
Fourier-Net	RadIal	12.9 G	327.68k
RD FFT	RadIal	214 M	N/A
Fourier-Net	RADDet	337.6 M	69.63k
RD FFT	RADDet	11 M	N/A

Table 5.11: **Computational overhead:** Floating point operations per second (FLOPs) and network parameters. FLOPs is equated as 2·FLOPs $\sim$ 1· MACs (multiply and accumulate operations). Note that models utilizing ADC inputs will be the summation of the Fourier-Net and T-FFTRadNet.

5.3 Architecture specifications and computing resources.

We utilize PyTorch *1.12.1* [50] as the deep learning framework. All code is built upon the framework provided by Rebut et al. [1] All training and inference are done utilizing a single Nvidia RTX 3090 24GB card and Intel ® i7 12700K CPU. Models are trained for 150 epochs total utilizing the Adam optimizer with an initial learning rate of 10^{-4} and a decay of 0.9 every 10 epochs. The optimal model is selected via an F1 score for the HD radar, and by maximizing performance over both heads (classification, general object detection) for LD radar. In both cases, this is done utilizing the validation dataset. For the HD radar, a batch size of 4 is utilized, where as the LD radar we utilize batches of size 16 on RD inputs. For the full, padded RAD cube, we utilize batches of size 4. Table 5.11 contains information regarding network complexity and computational requirements for a single inference. Values for our architecture and FFTRadNet are calculated utilizing the *DeepSpeed* framework [51], Cross Modal DNN is taken from [42] and RADDet is calculated using TensorFlow's profiling functions [52].

5.4 Object Detection on RadIal (HD)

Models are evaluated on the test set given in Rebut et al. [1], utilizing AP and AR for object detection, and mean intersection over union (mIoU) for free driving space segmentation. Table 5.12 shows a collection of results for comparison between other models on the same dataset. The table also contains various renditions of our implementation operating on different inputs, or different pre-processing, *i.e.*, windowing or no windowing. Results with the best performance are presented in bold.

	AP (↑)	AR(↑)	F1(↑)	Range Error(↓)	Angle Error(↓)	mIoU(↑)
FFTRadNet [1]	**96.8**%	82.2%	88.9%	**0.12m**	**0.10**°	73.98%
Cross Modal DNN [42]	**96.9**%	83.5%	**89.7**%	N/A	N/A	**80.4**%
T-FFTRadNet	89.6%	**89.5**%	**89.5**%	$0.15m$	0.12°	**80.2**%
T-FFTRadNet (Shift)	91.3%	88.3%	**89.8**%	$0.15m$	0.12°	79.5%
T-FFTRadNet (No Windowing)	89.4%	87.6%	88.5%	$0.16m$	0.12°	79.3%
T-FFTRadNet (ADC)	88.2%	86.7%	87.4%	$0.16m$	0.13°	79.6%
T-FFTRadNet (ADC - NL)	88.1%	86.1%	87.1%	$0.16m$	0.13°	78.8%
T-FFTRadNet (ADC - No Shift)	88.1%	85.5%	86.7%	$0.16m$	0.13°	78.8%

Table 5.12: **HD Radar - Architecture comparison:** Transformer based FFTRadNet (T-FFTRadNet), FFTRadNet [1] and Cross Modal DNN [42]. Labels in brackets denote differing input types, or the addition/removal of pre-processing steps, *i.e.*, removal of Hamming Window or utilizing an *fftshift*.

From Table 5.12, utilization of the Swin Transformer on the standard RadIal data provides significant increases in AR in comparison to FFTRadNet [1] and Cross Modal DNN [42] by $\sim 6-7\%$, although AP generally lacks by $\sim 7\%$. It is interesting to note the equivalence between the two metrics (AP and AR) with our architecture, resulting in an F1 score on par with the state-of-the-art (SOTA) and a more balanced model in general. This is also the case regarding free driving space evaluation in which the mIoU is significantly higher than FFTRadNet by $\sim 6\%$ and on par with Cross Modal DNN. We remind the reader that the Cross Modal DNN strategy utilizes improved labels. We also note that applying a Hamming window is an efficient method of improving performance and is consistent with results in later sections. Centering the zero frequency bin in the Doppler spectrum (denoted with Shift in Table 5.12) improves performance marginally in some metrics. It is likely not

a crucial pre-processing step given that the amount of spectral leakage with high-resolution data will be fairly low. Utilizing a shift should be investigated on a case-by-case basis.

Interestingly, we do not see increased performance via the utilization of raw ADC for HD radar. It is possible that in cases where resolution is not an issue, the utilization of a standard FFT is sufficient. Further optimization may yield parameter combinations that increase this performance. Being that network inputs are not normalized by design, the expected range is no longer static, which influences performance. We also note that the utilization of NL activation provides no benefit in our study and in fact, slightly inhibits performance likely due to the suppression of negative values. In either case, the transformation learned is no longer Fourier. The transformation contains a perturbation from a standard FFT.

5.5 Object Detection on RADDet (LD)

Models are evaluated on the test set given in Zhang et al. [4], in which the metric of choice is mean average precision (mAP). Table 5.13 contains a comparison of our implementation, utilizing various inputs and the published results of RADDet [4] and DAROD [43]. We utilize an IoU and NMS thresholds of 0.5 and 0.3, respectively, as used in Zhang et al.[4]

From Table 5.13, utilizing the transformer-based network on full RAD cubes significantly outperforms RADDet by $\sim 4\%$, and marginally outperforms DAROD utilizing RD inputs. We see a significant increase in performance over RD inputs when raw ADC is used, in terms of class-wise mAP (Table 5.13 top). The model is able to learn a more optimal transformation (a perturbed FFT), along with the sensor characteristics of the radar sensor. The result is a model more capable of classifying objects within the radar spectrum. For LD radar datasets with shorter sequences in range, azimuth and Doppler dimensions, the resulting frequency domain resolution, referring to the Fourier transform main lobe widths, will be more limited. Utilizing data-dependent NN models can perhaps learn the average SNR of the data across each frequency, and weight accordingly the information from each frequency, therefore providing a benefit that can compensate for the reduced resolution, up

RD and ADC Inputs

	T-FFTRadNet	T-FFTRadNet (NL)	T-FFTRadNet	T-FFTRadNet	Decourt et al.[43]
Input	ADC	ADC	RD	RD (No Windowing)	RD
mAP(↑)	**49.1**%	45.6%	46.8%	44.6%	46.6%

RAD Inputs

	T-FFTRadNet	Zhang et al.[4]
Input	RAD	RAD
mAP(↑)	**55.7**%	51.6%

Table 5.13: **LD Radar - Architecture comparison:** mean Average Precision over the six object classes. Sub-tables correspond to models in which complexity and input structure are the same, ADC and RD (top) and RAD cube (bottom).

to some point.

Generalized object detection results are presented in Table 5.15, utilizing the binary RA grid devised in FFTRadNet. The most optimal model with respect to general detection does not correspond to the highest mAP (all things equal), although the disparity between them is on the order of a few percent (ADC outperforms RD class-wise, RD outperforms in general detection). It is possible that loss contributions and their respective weighting play the reason for this discrepancy. Although both models utilize the same weightings, we have introduced additional transformation parameters with respect to ADC input. These parameters may be influenced in such a way that they better learn the differences between different object signatures, at the expense of general object feature extraction. Being that pixel-wise cross-entropy is subject to sparsity (large amounts of zeros), the loss is scaled in such a way that it is on equal orders of magnitude throughout training and therefore the trade-off should be partially minimized. Optimized loss weighting schemes, or perhaps the utilization of a different loss function could improve overall performance.

Table 5.14 contains class-wise AP and AR values for the models. From this table, it is apparent that the transformer network suffers specifically on the motorcycle class. This is due to the fact that the class is underrepresented within the dataset, causing these to be commonly classified as bicycles or cars depending on their relative position and angle with respect to the sensor.

Model	$\mathbf{AP}_{bicycle}$	$\mathbf{AP}_{bus}$	$\mathbf{AP}_{car}$	$\mathbf{AP}_{motorcycle}$	$\mathbf{AP}_{person}$	$\mathbf{AP}_{truck}$
T-FFTRadNet (ADC - NL)	56.6%	37.5%	62.6%	0.0%	66.5%	50.5%
T-FFTRadNet (ADC)	62.5%	**50.0**%	63.0%	0.0%	70.7%	48.4%
T-FFTRadNet (RD)	**72.6**%	20.0%	**65.1**%	0.0%	**71.8**%	51.4%
T-FFTRadNet (RD - No Windowing)	62.3%	28.6%	60.6%	0.0%	70.6%	45.7%
T-FFTRadNet (RAD)	60.1%	**50.0**%	64.6%	**33.3**%	70.6%	**55.9**%
	$\mathbf{AR}_{bicycle}$	$\mathbf{AR}_{bus}$	$\mathbf{AR}_{car}$	$\mathbf{AR}_{motorcycle}$	$\mathbf{AR}_{person}$	$\mathbf{AR}_{truck}$
T-FFTRadNet (ADC - NL)	29.4%	7.9%	43.6%	0.0%	36.9%	30.4%
T-FFTRadNet (ADC)	31.9%	10.5%	44.5%	0.0%	38.8%	31.7%
T-FFTRadNet (RD)	29.9%	5.3%	**50.6**%	0.0%	39.6%	35.4%
T-FFTRadNet (RD - No Windowing)	21.1%	5.3%	37.5%	0.0%	33.5%	27.1%
T-FFTRadNet (RAD)	**34.8**%	**18.4**%	50.5%	**4.8**%	**44.9**%	**39.9**%

Table 5.14: **LD Radar - Classwise AP and AR:** Notice the decrease in performance across specific classes. The cause of this is twofold, the first reason being that certain classes are underrepresented in the dataset, and the second is due to similar signatures in the RD spectrum from certain objects.

Architecture	Input	AP (↑)	AR (↑)
T-FFTRadNet	ADC (NL)	59.9%	54.1%
T-FFTRadNet	ADC	61.8%	54.5%
T-FFTRadNet	RD	**64.5**%	**57.5**%
T-FFTRadNet	RD (No Windowing)	58.5%	48.4%
T-FFTRadNet	RAD	**64.2**%	**56.5**%

Table 5.15: **LD Radar - Generalized Detection Head Results:** Results from the generalized object detection head.

5.6 Network Range Perception

A critical issue with radar sensors is the degradation of performance, in terms of received power, as an object's radial distance increases. Given that this issue directly effects the output spectra, this issue can also persist in the NNs used to analyze the data in tasks such as object detection. Fig. 5.7 depicts the network performance in terms of general object detection (AP and AR) as a function of range for both HD (top) and LD (bottom) radar datasets. Different input types are indicated via the text in the top left corner of the plots. Grey bars indicate the fraction of total objects within a given range bin. Blue and red bars denote AR and AP, respectively. We apply a minimum distance threshold of $5m$ in each case to alleviate reflection pileup, for example off the hood of the vehicle itself. We provide an

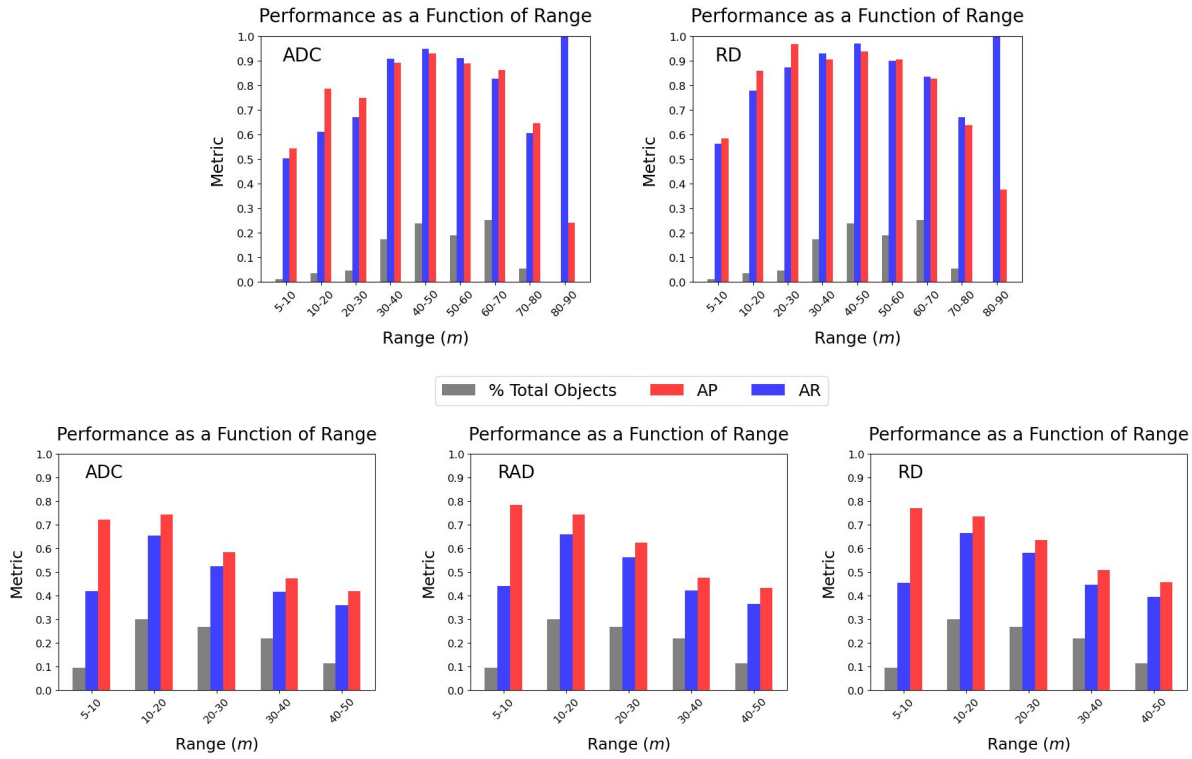


Figure 5.7: **General object detection performance as a function of range:** AP and AR as a function of discrete range bins for HD (top) and LD (bottom) radar datasets. Input type is indicated via the text in the top left corner of the plots. Grey represents the fraction of total objects in the test set within a specific range bin, blue and red denote AR and AP respectively. A minimum distance threshold of $5m$ is applied to both datasets given there is often large amounts of noise due to the location of the sensor on the front of the vehicle.

upper threshold of $90m$ for the HD radar dataset, and $50m$ for the LD radar dataset based off the most distant object in the testing dataset.

HD radar produces a performance distribution that is more flat as a function of range in comparison to LD radar, in which one must take into consideration the uncertainties attributed to relatively low amounts of data in certain range bins. Peak performance occurs within center of the distribution in which the tails (both closer to, and further away from the sensor) exhibit decreased performance, although the model is still able to reliably detect objects across all ranges. In the case of LD radar, we see a clear decline in performance as function of range, which is to be expected given the relatively low number of T_x and R_x. The power received from distant objects will decrease significantly as a function of range

and therefore the quality of network performance will also degrade in these regions.

5.7 Network Angular Perception

As discussed in Sec. 2.1, the angular resolution of a radar sensor is not linear and instead follows a sinusoidal relation. That is to say, the side lobe profile of an objects frequency peak changes as a function of angle in which objects at larger angles may be harder to resolve as individual peaks. We investigate the models performance as a function of angle, in which we note the FOV of both the sensor and our algorithm is 90°. The distribution of objects in the tests sets do not encompass the FOV, in which the HD radar test set only provides objects in the range of ±30° and the LD radar test provides objects in the range of ±60°. Fig. 5.8 depicts the network performance as a function of angle for both the HD (top) and LD (bottom) datasets, in which we utilize general object detection metrics (AP and AR) as in the section prior.

From the figure, the HD radar depicts a performance distribution similar to what is seen as a function of range, meaning the degree of flatness of the performance distribution as a function of angle, although the distributions are not symmetrical. Ideally the network should be ambiguous to positive or negative angles, however due to low amounts of data at larger angles it is somewhat hard to make a conclusive decision as to whether or not this is the case. When uncertainties are considered it is likely the distributions are symmetric. It also is apparent that the RD inputs outperform ADC inputs to a degree across some angular bins. In the case of LD radar, we see a similar pattern of lacking symmetry around positive and negative angles, although when uncertainties are considered the symmetry is again reasonable. We also note the flatness of the distributions, in which it is apparent the model is capable of performing object detection across all regions of the phase space.

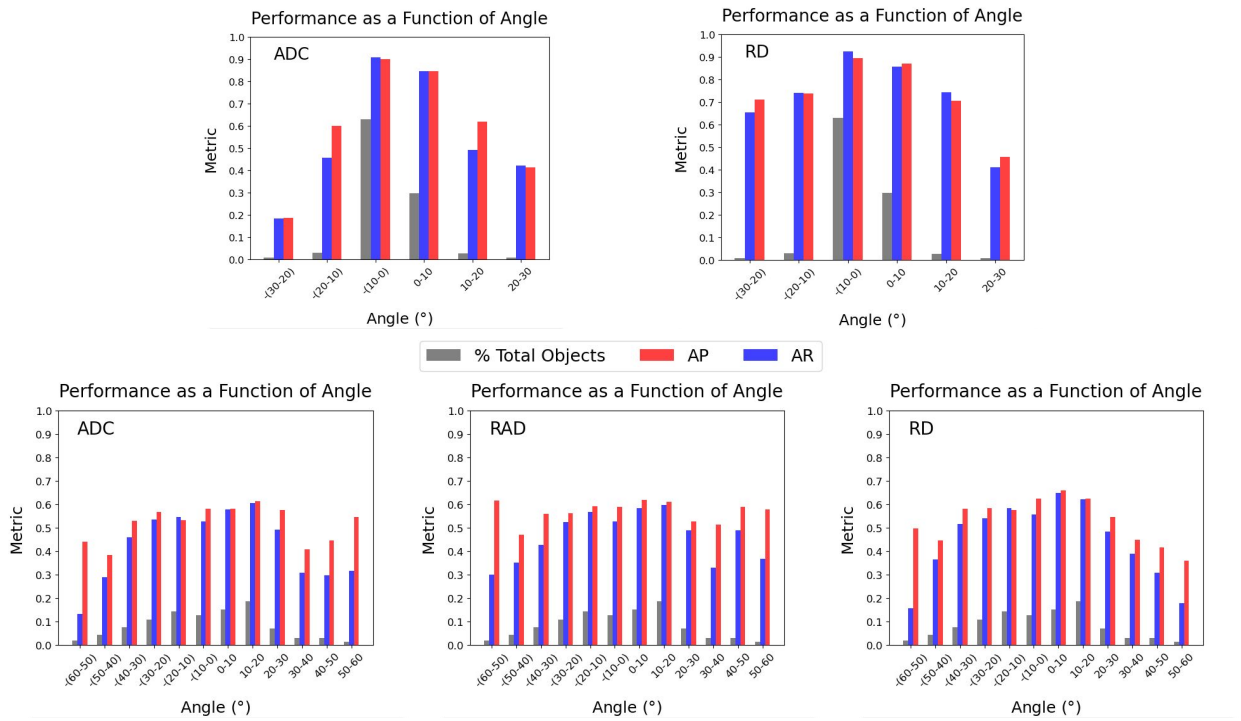


Figure 5.8: **General object detection performance as a function of angle:** AP and AR as a function of discrete angle bins for HD (top) and LD (bottom) radar datasets. Input type is indicated via the text in the top left corner of the plots. Grey represents the fraction of total objects in the test set within a specific range bin, blue and red denote AR and AP respectively. The FOV of the sensors and the network is 90°, however the testing set distribution does not cover the entire space.

5.8 Example Annotations

In this section we provide example annotations from our network which have been translated into the image domain. In what follows, we will show instances of both exceptional and poor performance and discuss some of the potential reasons for the networks failure in the scenes shown in Fig. 5.9.The ground truth boxes will be depicted in blue, and predicted boxes will be depicted in red, both of which are overlayed onto the camera frame in the left side of images. On the right hand side of images, we depict the regressed free driving space of the network in the radar frame (polar coordinate system) in which the bounding boxes are overlayed in a similar fashion.

From inspection of the above figure (top three rows), it can be seen that the model has

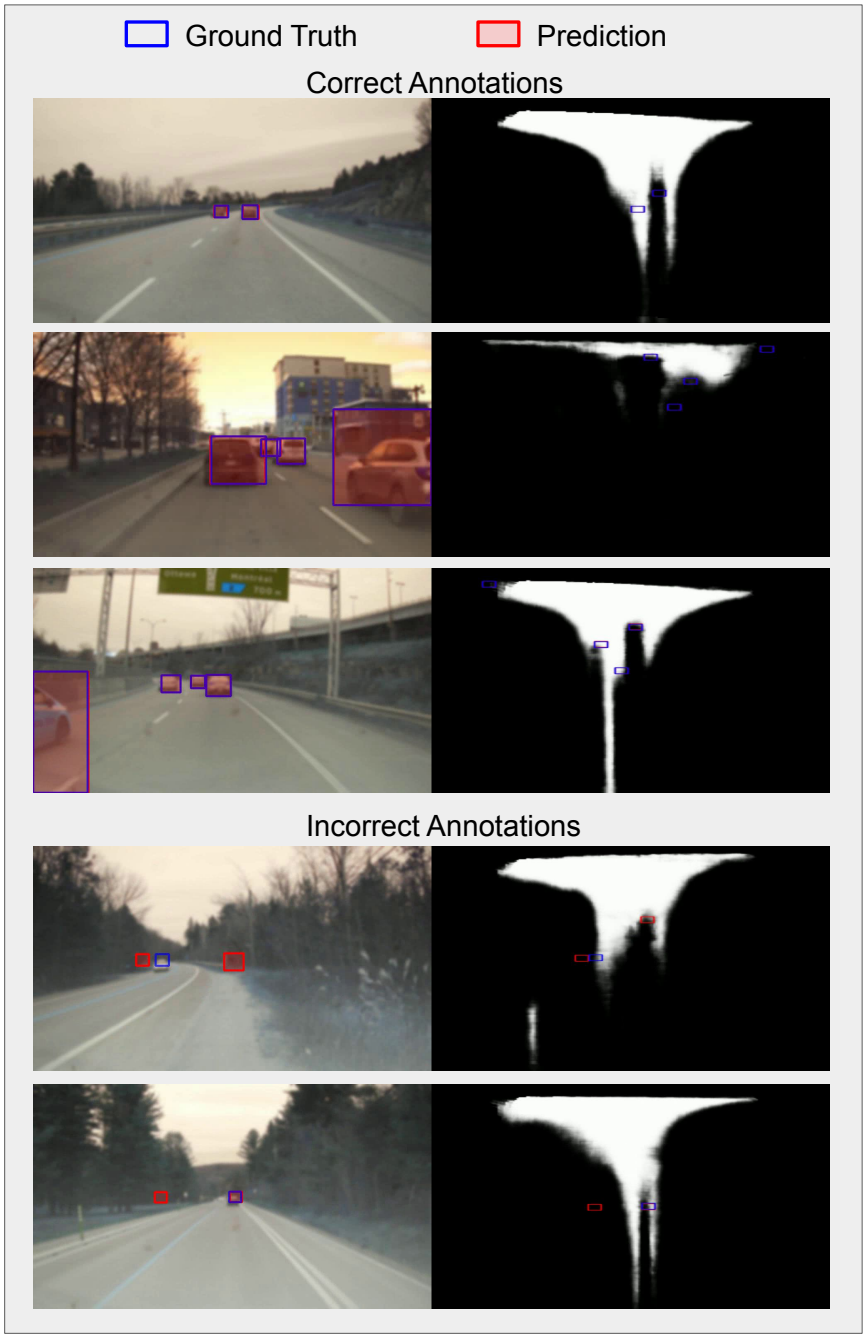


Figure 5.9: **T-FFTRadNet example annotations:** Annotations from T-FFTRadNet operating on radar data translated to the camera frame (left). Free driving space segmentation in polar coordinates with overlayed annotations. Note in some cases the predicted and ground truth boxes overlap to a very high degree in the segmentation image. Instances of exceptional model performance are provided in the top three images, instances of poor performance are shown in the bottom two images.

learned to efficiently detect vehicles in a multitude of different driving conditions. We provide instances of highway driving with relatively large amount of spacing between the sensor and the object, along with more complex downtown and freeway scenes in which the number of objects is much higher. Interestingly, in can be seen in the second from bottom image that the annotation of the vehicle on the left side of the image is shifted leftward. It is likely the reason for this is due to the heavy amount of background objects (such as trees) providing a steady return rate. This makes it harder for the model to characterize the vehicles response above background. We can also see in this image that the network has identified the end of the guard rail as an object that the vehicle should consider. Although this is not a true annotation, it is likely a useful object for the network to detect for downstream tasks. Looking at the last image we see correct annotation of the vehicle on the road, although we have a false detection due to a heavy presence of background.

5.9 Discussion

We have shown that Vision Transformers, namely hierarchical Swin Transformers are capable of forming the backbone for efficient object detection on varying radar inputs. Given that transformers utilize primarily dense connections and embeddings, the computational throughput required for a single inference pass is significantly lower (see FLOPs in Table 5.11) than convolutional based feature extractors. Moreover, the adaptation to differing inputs (RD vs RAD for example) requires fewer network alterations given fixed-sized embeddings.

Utilizing linear layers to mimic FFT response is an efficient method of decreasing pre-processing costs, while retaining performance on par or potentially better, depending on the task, compared to traditional DSP pre-processing and normalization. The layers are able to better encapsulate the radar sensor characteristics in this case. This is encapsulated within the LD radar task of object classification, in which we see utilization of raw ADC and the linear layers provides an increased mAP. In HD radar, utilizing traditional RD inputs performs best, although there is not a significant performance decrease when ADC is utilized.

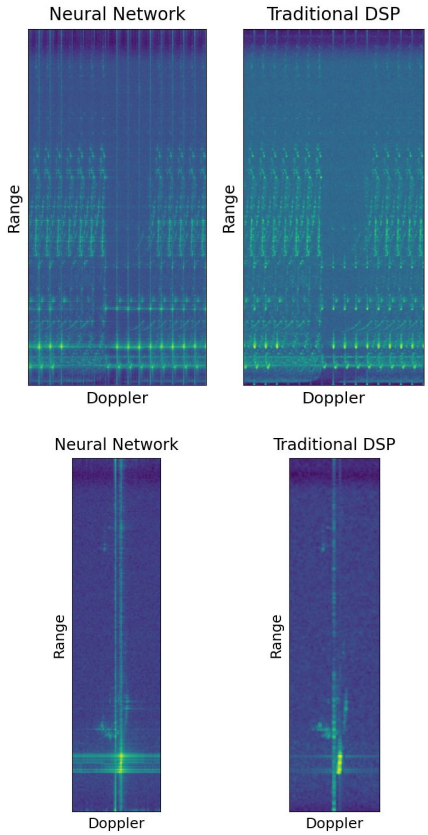


Figure 5.10: **Fourier-Net transformation comparison:** Learned transformations (left column) and traditional RD transformations utilizing Hamming Windows, FFTs, and centering of zero Doppler frequency (right column). HD radar (top row) and LD radar (bottom row).

In either case, we note that the transformation learned is no longer Fourier. Fig. 5.10 shows traditional RD FFTs (right column) along with the learned transformation (left column) for both HD (top row) and LD (bottom row) radar.

Being that the Fourier layers can be optimally utilized on GPU, the inference time for a single event decreases in comparison to utilizing RD inputs. If one takes into account the necessary time needed to perform standard FFTs and normalization for RD inputs on HD radar, the average time per one inference was $\sim 156ms$ in our implementation, whereas with raw ADC the value was one-fifth of this at $\sim 30ms$. LD radar is similar, with the average time per one inference at $\sim 152ms$ with RAD cubes, $\sim 33ms$ with RD inputs, and $\sim 20ms$ with raw ADC inputs. Thus, the utilization of these layers allows the transition

toward real-time applications. Note the above values were calculated utilizing an Nvidia RTX 3090 24GB card and Intel ® i7 12700K CPU in which FFTs are allocated to the CPU. Performing an FFT utilizing a CPU is likely similar, in terms of time, to that of our transformation network running on GPU in which the bulk of time savings occurs from the removal of normalization, shifting indices (*fftshift*) and reordering complex components as channels.

Hamming windows have shown increased performance in both LD and HD radar datasets due to their ability to reduce side lobe profiles. Centering the zero Doppler frequency, via an *fftshift*, produces performance increases for LD radar, but performance remains approximately the same for HD radar. Due to higher resolution, leakage across frequencies does not pose the same impacts as seen in LD radar, where the centering increases performance significantly.

Chapter 6

Conclusion and Future Work

Radar possesses many desirable traits in relation to autonomous vehicles and other robotic applications, specifically those that rely on the localization of objects in space. The capability to operate in adverse conditions reliably, unlike other emission-based sensors such as LiDAR, make radar a must for fully autonomous systems. Traditional methods of utilizing radar point clouds fall short due to relative sparseness in the generated clouds. To overcome this, methods coupling traditional DSP methods combined with deep learning have been proposed. This too poses new problems, such as the relative computational costs associated with pre-processing inputs utilizing FFTs along with the network growth as input complexity scales. In this work we have shown that we can operate on par, or better depending on the task, utilizing raw ADC inputs via the utilization of fully connected layers that mimic traditional FFTs, yet utilize the efficient and fast computational power of GPUs. These layers require no pre-processing of inputs (including normalization) and can potentially allow the model to learn the physical characteristics of the sensor. This consideration seems to be less prominent in HD radar, where resolution is not an issue. The utilization of dense transformation layers greatly decreases the time associated with a single inference given that no normalization or FFTs are required. Moreover, in its current semi-optimized state, the model utilizing raw ADC performs only slightly worse ($\tilde{1}.5\%$ decrease in F1 score) yet can be deployed in real time.

We show the novel application of vision transformers, namely hierarchical Swin Transformers [2] to radar-based object detection, utilizing various inputs. Swin transformers utilize W-MHSA which allows the model to retain linear complexity with respect to the input, a strong consideration when deployment on relatively low-power hardware is considered. It is also known that vision transformers scale with training data and as such our contribution is timely given the growing interest in the generation of radar object detection datasets. Our architecture builds off the prior work of Rebut et al. [1], and is capable of obtaining SOTA results across various datasets with differing sensor configurations, *i.e.*, LD and HD radar. The model primarily takes ADC or RD matrices as input and approximates the angular transformation required for object detection, but we have also shown the capability to operate on full RAD cubes with little network alterations, mainly due to fixed-sized embeddings utilized by transformers. This also promotes minimal network growth.

6.1 Future Work

In its current state, our architecture has shown increased performance across varying tasks and datasets. However, the model is far from optimal and could be further improved via redefinition of the decoder to allow better feature retention across the axis transformation. Ideally, one could construct a transformer based decoder capable of minimizing the number of axis permutations to reduce information loss seen by the Swin decoders. Given the results utilizing a transformer as the encoder, we expect improved results can be obtained. Moreover, the utilization of a transformer decoder structure would further help reduce FLOPs given the utilization of only dense layers as opposed to convolutions, a critical consideration when deploying to real time devices. In any case, the model should also be subject to a rigorous hyperparameter optimization scheme as it is possible there exists more optimal states in terms of both complexity reduction and performance. Specifically, the model utilizing raw ADC input should be optimized to further improve performance given the minimal amount of time required to perform inference.

To better show the networks generalizability we would like to utilize other radar datasets. Currently, only a single HD radar dataset is available and has been utilized. In this case, information from the camera could be used to provide annotations for objects within the dataset, allowing the classification performance of the network to be tested on the HD radar dataset. For LD radar, we wish to provide performance benchmarks on various inputs for the recently created Pixset dataset [53]. This dataset is both larger and more difficult than traditional LD radar datasets, containing dense urban scenes. Given that the density of objects is high in such a scene, the dataset is ideal as a method of stress testing the reliability of radar object detection models utilizing LD radar sensors. However, in utilizing the dataset we have encountered unforeseen issues that are difficult to trace to their origins. Initially, the issues seemed to persist with the translation of annotations from LiDAR to radar reference frames. To circumvent this, Huet et al.[54] devised an annotation scheme utilizing the clustering of projected LiDAR points into a segmented image frame. These points were annotated utilizing the camera and reprojected into the original cartesian coordinate system. This approach provided accurate and reliable annotations, yet issues with performance persisted. As a result, we believe there exists an issue with the radar data itself and requires further investigation to be used. In any case, further investigation into the model's performance as a function of object velocity should be undertaken. Given a specific radar sensor configuration, an objects velocity directly affects a network's ability to correctly detect and classify objects within the radar frame.

Bibliography

[1] J. Rebut, A. Ouaknine, W. Malik, and P. Pérez, "Raw High-Definition Radar for Multi-Task Learning," 2021. [Online]. Available: https://arxiv.org/abs/2112.10646

[2] Z. Liu, Y. Lin, Y. Cao, H. Hu, Y. Wei *et al.*, "Swin Transformer: Hierarchical Vision Transformer Using Shifted Windows," in *Proceedings of the IEEE CVF International Conference on Computer Vision (ICCV)*, October 2021, pp. 10 012–10 022.

[3] P. Zhao, C. X. Lu, B. Wang, N. Trigoni, and A. Markham, "CubeLearn: End-to-end Learning for Human Motion Recognition from Raw mmWave Radar Signals," *IEEE Internet of Things Journal*, pp. 1–1, 2023.

[4] A. Zhang, F. E. Nowruzi, and R. Laganiere, "RADDet: Range-Azimuth-Doppler based Radar Object Detection for Dynamic Road Users," in *2021 18th Conference on Robots and Vision (CRV)*, 2021, pp. 95–102.

[5] A. Manikas, "EE3-27: Principles of Classical and Modern Radar," February 2020.

[6] C. Iovescu and S. Rao, "The fundamentals of millimeter wave sensors," *Texas Instruments*, pp. 1–8, 2017.

[7] R. Yamashita, M. Nishio, R. K. G. Do, and K. Togashi, "Convolutional neural networks: an overview and application in radiology," *Insights into imaging*, vol. 9, pp. 611–629, 2018.

[8] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai *et al.*, "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale," 2021. [Online]. Available: https://arxiv.org/abs/2010.11929

[9] Y. Li, J. Wang, X. Dai, L. Wang, C.-C. M. Yeh *et al.*, "How Does Attention Work in Vision Transformers? A Visual Analytics Attempt," 2023. [Online]. Available: https://arxiv.org/abs/2303.13731

[10] P. Lippe, "Tutorial 6: Transformers and Multi-Head Attention," March 2023. [Online]. Available: https://uvadlc-notebooks.readthedocs.io/en/latest/tutorial_notebooks/tutorial6/Transformers_and_MHAttention.html

[11] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks," in *Advances in Neural Information Processing Systems*, C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett, Eds., vol. 28. Curran Associates, Inc., 2015. [Online]. Available: https://proceedings.neurips.cc/paper/2015/file/14bfa6bb14875e45bba028a21ed38046-Paper.pdf

[12] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," 2013. [Online]. Available: https://arxiv.org/abs/1311.2524

[13] R. Girshick, "Fast R-CNN," in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, December 2015.

[14] K. He, G. Gkioxari, P. Dollar, and R. Girshick, "Mask R-CNN," in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, Oct 2017.

[15] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," 2015. [Online]. Available: https://arxiv.org/abs/1512.03385

[16] K. Simonyan and A. Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition," 2014. [Online]. Available: https://arxiv.org/abs/1409.1556

[17] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks," in *Advances in Neural Information Processing Systems*, F. Pereira, C. Burges, L. Bottou, and K. Weinberger, Eds., vol. 25. Curran Associates, Inc., 2012. [Online]. Available: https://proceedings.neurips.cc/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf

[18] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan *et al.*, "Feature Pyramid Networks for Object Detection," 2016. [Online]. Available: https://arxiv.org/abs/1612.03144

[19] S. Xie, R. Girshick, P. Dollár, Z. Tu, and K. He, "Aggregated Residual Transformations for Deep Neural Networks," 2016. [Online]. Available: https://arxiv.org/abs/1611.05431

[20] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection," 2015. [Online]. Available: https://arxiv.org/abs/1506.02640

[21] J. Redmon and A. Farhadi, "YOLO9000: Better, Faster, Stronger," 2016. [Online]. Available: https://arxiv.org/abs/1612.08242

[22] J. Redmon and A. Farhadi, "YOLOv3: An Incremental Improvement," 2018. [Online]. Available: https://arxiv.org/abs/1804.02767

[23] A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao, "YOLOv4: Optimal Speed and Accuracy of Object Detection," 2020. [Online]. Available: https://arxiv.org/abs/2004.10934

[24] J. Redmon, "Darknet: Open source neural networks in c," http://pjreddie.com/darknet/, 2013–2016.

[25] S. Liu, L. Qi, H. Qin, J. Shi, and J. Jia, "Path Aggregation Network for Instance Segmentation," 2018. [Online]. Available: https://arxiv.org/abs/1803.01534

[26] M. Tan, R. Pang, and Q. V. Le, "EfficientDet: Scalable and Efficient Object Detection," 2019. [Online]. Available: https://arxiv.org/abs/1911.09070

[27] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed *et al.*, "SSD: Single shot MultiBox detector," in *Computer Vision – ECCV 2016*. Springer International Publishing, 2016, pp. 21–37. [Online]. Available: https://doi.org/10.1007%2F978-3-319-46448-0_2

[28] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation," 2016. [Online]. Available: https://arxiv.org/abs/1612.00593

[29] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, "PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space," 2017. [Online]. Available: https://arxiv.org/abs/1706.02413

[30] J. Beltrán, C. Guindel, F. M. Moreno, D. Cruzado, F. García *et al.*, "BirdNet: A 3D Object Detection Framework from LiDAR Information," in *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, 2018, pp. 3517–3523.

[31] Y. Guo, H. Wang, Q. Hu, H. Liu, L. Liu *et al.*, "Deep Learning for 3D Point Clouds: A Survey," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 43, no. 12, pp. 4338–4364, 2021.

[32] Z. Liang and Y. Huang, "Survey on deep learning-based 3D object detection in autonomous driving," *Transactions of the Institute of Measurement and Control*, vol. 45, no. 4, pp. 761–776, 2023. [Online]. Available: https://doi.org/10.1177/01423312221093147

[33] P. Svenningsson, F. Fioranelli, and A. Yarovoy, "Radar-PointGNN: Graph Based Object Recognition for Unstructured Radar Point-cloud Data," in *2021 IEEE Radar Conference (RadarConf21)*, 2021, pp. 1–6.

[34] R. Nabati and H. Qi, "CenterFusion: Center-Based Radar and Camera Fusion for 3D Object Detection," in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, January 2021, pp. 1527–1536.

[35] L. Wang, T. Chen, C. Anklam, and B. Goldluecke, "High Dimensional Frustum PointNet for 3D Object Detection from Camera, LiDAR, and Radar," in *2020 IEEE Intelligent Vehicles Symposium (IV)*, 2020, pp. 1621–1628.

[36] R. Prophet, A. Deligiannis, J.-C. Fuentes-Michel, I. Weber, and M. Vossiek, "Semantic Segmentation on 3D Occupancy Grids for Automotive Radar," *IEEE Access*, vol. 8, pp. 197 917–197 930, 2020.

[37] A. Danzer, T. Griebel, M. Bach, and K. Dietmayer, "2D Car Detection in Radar Data with PointNets," in *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*, 2019, pp. 61–66.

[38] M. Meyer and G. Kuschk, "Deep Learning Based 3D Object Detection for Automotive Radar and Camera," in *2019 16th European Radar Conference (EuRAD)*, 2019, pp. 133–136.

[39] A. Palffy, J. Dong, J. F. P. Kooij, and D. M. Gavrila, "CNN Based Road User Detection Using the 3D Radar Cube," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 1263–1270, 2020.

[40] X. Gao, G. Xing, S. Roy, and H. Liu, "RAMP-CNN: A Novel Neural Network for Enhanced Automotive Radar Object Recognition," *IEEE Sensors Journal*, vol. 21, no. 4, pp. 5119–5132, 2021.

[41] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2015: 18th International Conference, Munich, Germany, October 5-9, 2015, Proceedings, Part III 18*. Springer, 2015, pp. 234–241.

[42] Y. Jin, A. Deligiannis, J.-C. Fuentes-Michel, and M. Vossiek, "Cross-Modal Supervision-Based Multitask Learning with Automotive Radar Raw Data," *IEEE Transactions on Intelligent Vehicles*, pp. 1–15, 2023.

[43] C. Decourt, R. VanRullen, D. Salle, and T. Oberlin, "DAROD: A Deep Automotive Radar Object Detector on Range-Doppler maps," in *2022 IEEE Intelligent Vehicles Symposium (IV)*, 2022, pp. 112–118.

[44] A. Ouaknine, A. Newson, P. Pérez, F. Tupin, and J. Rebut, "Multi-View Radar Semantic Segmentation," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2021, pp. 15 671–15 680.

[45] N. Shirakata, K. Iwasa, T. Yui, H. Yomo, T. Murata *et al.*, "Object and Direction Classification based on Range-Doppler Map of 79 GHz MIMO Radar Using a Convolutional Neural Network," in *2019 12th Global Symposium on Millimeter Waves (GSMM)*, 2019, pp. 1–3.

[46] A. Ouaknine, A. Newson, J. Rebut, F. Tupin, and P. Pérez, "CARRADA Dataset: Camera and Automotive Radar with Range- Angle- Doppler Annotations," in *2020 25th International Conference on Pattern Recognition (ICPR)*, 2021, pp. 5068–5075.

[47] J. Long, E. Shelhamer, and T. Darrell, "Fully Convolutional Networks for Semantic Segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015.

[48] L.-C. Chen, G. Papandreou, F. Schroff, and H. Adam, "Rethinking Atrous Convolution for Semantic Image Segmentation," 2017. [Online]. Available: https://arxiv.org/abs/1706.05587

[49] M. Arjovsky, A. Shah, and Y. Bengio, "Unitary Evolution Recurrent Neural Networks," in *Proceedings of The 33rd International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, M. F. Balcan and K. Q. Weinberger, Eds., vol. 48. New York, New York, USA: PMLR, 20–22 Jun 2016, pp. 1120–1128. [Online]. Available: https://proceedings.mlr.press/v48/arjovsky16.html

[50] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury *et al.*, "PyTorch: An Imperative Style, High-Performance Deep Learning Library," in *Advances in Neural Information Processing Systems*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, Eds., vol. 32. Curran Associates, Inc., 2019. [Online]. Available: https://proceedings.neurips.cc/paper/2019/file/bdbca288fee7f92f2bfa9f7012727740-Paper.pdf

[51] J. Rasley, S. Rajbhandari, O. Ruwase, and Y. He, "DeepSpeed: System Optimizations Enable Training Deep Learning Models with Over 100 Billion Parameters," in *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, ser. KDD '20. New York, NY, USA: Association for Computing Machinery, 2020, p. 3505–3506. [Online]. Available: https://doi.org/10.1145/3394486.3406703

[52] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen *et al.*, " TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems," 2015, software available from tensorflow.org. [Online]. Available: https://www.tensorflow.org/

[53] J.-L. Déziel, P. Merriaux, F. Tremblay, D. Lessard, D. Plourde *et al.*, "PixSet : An Opportunity for 3D Computer Vision to Go Beyond Point Clouds With a Full-Waveform

LiDAR Dataset," 2021. [Online]. Available: https://arxiv.org/abs/2102.12010

[54] A. Huet, Private Communication.

[55] H. Cao, Y. Wang, J. Chen, D. Jiang, X. Zhang *et al.*, "Swin-Unet: Unet-like Pure Transformer for Medical Image Segmentation," 2021. [Online]. Available: https://arxiv.org/abs/2105.05537

[56] J.-L. Déziel, P. Merriaux, F. Tremblay, D. Lessard, D. Plourde *et al.*, "PixSet : An Opportunity for 3D Computer Vision to Go Beyond Point Clouds With a Full-Waveform LiDAR Dataset," 2021. [Online]. Available: https://arxiv.org/abs/2102.12010

[57] S. Scardapane, S. Van Vaerenbergh, A. Hussain, and A. Uncini, "Complex-Valued Neural Networks With Nonparametric Activation Functions," *IEEE Transactions on Emerging Topics in Computational Intelligence*, vol. 4, no. 2, pp. 140–150, 2020.