SCHOOL OF COMPUTATION, INFORMATION AND TECHNOLOGY - INFORMATICS

TECHNISCHE UNIVERSITÄT MÜNCHEN

Masters's Thesis in Informatics

Using Deep Learning for Estimation of Objects Properties with Radar in Low Earth Orbits

Mustafa Fatih Baysan

SCHOOL OF COMPUTATION, INFORMATION AND TECHNOLOGY - INFORMATICS

TECHNISCHE UNIVERSITÄT MÜNCHEN

Masters's Thesis in Informatics

Using Deep Learning for Estimation of Objects Properties with Radar in Low Earth Orbits

Objektcharakterisierung in Erdnahen Umlaufbahnen durch Deep Learning und Radar

Author:	Mustafa Fatih Baysan
Supervisor:	Prof. Hanz-Joachim Bungartz
Advisor:	M.Sc. Daniil Zverev & M.Sc. Fabio Gratl
Submission Date:	17.06.2024

I confirm that this masters's thesis in informatics is my own work and I have documented all sources and material used.

Munich, 17.06.2024 Mustafa Fatih Baysan

Acknowledgments

I would like to express my sincere gratitude to all those who have supported me throughout my journey in completing this thesis.

Firstly, I would like to thank my supervisor, Prof. Hanz-Joachim Bungartz, for making this thesis possible. And believing in the project.

I am also profoundly grateful to my advisors, M.Sc. Fabio Gratl and M.Sc. Daniil Zverev. Your unwavering support, constructive feedback, and never-ending patience under pressure have greatly improved the quality of my research. Your guidance and belief in my abilities, even during the most challenging times, have been invaluable. I deeply appreciate your commitment to my success and your dedication to fostering an environment of growth and learning.

To my family, especially my parents, who have been a constant source of support and inspiration, I am forever grateful. A heartfelt thank you to my brother, who has guided me through my academic career since the beginning and has been there for me every step of the way. Your unwavering belief in me has been my greatest motivation, and I owe my career to your constant guidance and support.

Finally, I would like to extend my heartfelt gratitude to my dear friend Abdi Umut Ege for his unwavering support, even when he couldn't be physically by my side. Throughout this journey, I have consistently felt your encouragement, and your presence has meant.

A special thank you goes to my colleagues and friends in the Actlabs. Your camaraderie, advice, and support have been crucial in making this journey both productive and enjoyable. I particularly appreciate the stimulating discussions and the collaborative environment that we have fostered together.

Thank you all for your support, without which this thesis would not have been possible.

Abstract

Radar technology has seen burgeoning applications across diverse domains, including automotive industries and surveillance systems, thereby intertwining with everyday life. Concurrently, the escalating challenge posed by space debris underscores the critical need for innovative solutions. Ground-based radars have historically addressed aspects of this challenge; however, airborne platforms are gaining prominence. Yet, the scarcity of publicly available data impedes rapid advancements in this field. Harnessing the power of deep learning presents a promising avenue for enhancing radar capabilities, potentially revolutionizing object recognition and predictive accuracy beyond traditional methodologies. This study introduces two novel data generation methods aimed at mitigating data scarcity and explores the transformative potential of deep learning in radar-based object recognition and prediction.

Kurzfassung

Die Popularität von Radarsystemen in verschiedenen Bereichen nimmt zu. Die breite Anwendung von Radartechnologie in Branchen wie der Automobilindustrie und Überwachungssystemen verbindet sie eng mit dem täglichen Leben. Ein weiteres aufkommendes Problem im Zusammenhang mit Raumfahrttechnologien ist die Bedrohung durch Weltraummüll. Radarsysteme besitzen das Potenzial, zur Lösung dieses Problems beizutragen. Obwohl bodengestützte Radarsysteme seit vielen Jahren für diesen Zweck eingesetzt werden, erfreuen sich luftgestützte Lösungen zunehmender Beliebtheit. Die Knappheit öffentlich zugänglicher Daten behindert jedoch die Beschleunigung von Fortschritten erheblich. Darüber hinaus kann Deep Learning Verbesserungen bringen und zu einer besseren Leistung führen. In dieser Arbeit wird das Ziel verfolgt, zwei verschiedene Methoden zur Datengenerierung vorzustellen, um der Knappheit an Daten entgegenzuwirken. Dabei wird Deep Learning genutzt, um das Potenzial zur Erkennung verschiedener Objektformen und deren Vorhersage im Vergleich zu konventionellen Ansätzen aufzuzeigen.

Contents

Acknowledgments iii

Abstract iv

Kurzfassung v

1 Introduction 1
- 1.1 Problem Definition . . . 1
  - 1.1.1 Scientific Challenges . . . 2
  - 1.1.2 Technical Challenges . . . 2
- 1.2 Motivation . . . 3
- 1.3 Radar Principles . . . 4
- 1.4 Radar Setup . . . 6

2 Literature Review 8
- 2.1 Radar Space Debris Detection . . . 8
- 2.2 Automotive Industry Solutions . . . 9
  - 2.2.1 Raw Signal Solutions . . . 10
  - 2.2.2 Complex Data Solutions . . . 11

3 Data Generation Pipeline 13
- 3.1 Shapenet . . . 13
- 3.2 Introduction to the Watertight Concept . . . 15
  - 3.2.1 Trimesh . . . 16
  - 3.2.2 Manifold and ManifoldPlus . . . 17
  - 3.2.3 Alternative Sub-optimal Approaches . . . 17
- 3.3 Radar Cross Section Calculation . . . 18
  - 3.3.1 Matlab Approach with Parallel Pool . . . 19
  - 3.3.2 Optix RCS . . . 21
  - 3.3.3 RaytrAMP: A Powerful RCS Calculation Tool . . . 22
  - 3.3.4 POfacets . . . 23
  - 3.3.5 FEKO Altair . . . 23
  - 3.3.6 Comparison and Performance . . . 24
- 3.4 Analysis of RCS output as slices and 3D . . . 25
- 3.5 Simulating Radar Return . . . 29
  - 3.5.1 Slice Approach . . . 29
  - 3.5.2 phased.BackscatterRadarTarget Function for Swerling Models . . . 30

3.5.3 Prior Actlabs Development 31

4 Deep Learning Models **32**
4.1 Used Architectures and Training Environment 32
4.2 Alternative Dataset Trials During the Test of Deep-Learning Models 34
4.2.1 Implementation of Deep Learning Environment 36
4.3 Obtained Results and Comparison 37

5 Complex Data Generation Simulation **40**
5.1 Simulation and Radar Parameters 41
5.2 Working Principle of Environment and Radar 42
5.3 Variables Defined in Simulation 45
5.4 Assumptions and Constraints 49
5.5 Filtered Matching and FFT Comparison 50
5.6 Output: IQ Signal and Pulse Doppler Image 52
5.7 Filtration of Output 56

6 Conclusion **58**
6.1 Findings 58
6.2 Possible Future Research 60

List of Figures **62**

List of Tables **64**

Bibliography **65**

1 Introduction

The rapid advancement of space technology has fueled a surge in satellite use, with launches exhibiting a logarithmic increase [1]. This exponential growth, driven by competition between nations and private companies, reflects the vital role satellites play in communication, navigation, Earth observation, and scientific research [2]. However, the harsh environment of space poses significant challenges to these critical devices. Any malfunction or disruption, internal or external, can render a satellite inoperable. While meticulous design and rigorous ground-based simulations can minimize internal failures, external threats remain difficult to predict and mitigate.

Among these external threats, space debris – a growing population of defunct spacecraft fragments, micrometeoroids, and other orbiting objects – poses a significant and increasing risk. Collisions with even small debris particles can cause catastrophic damage to operational satellites, leading to mission loss, cascading effects on dependent infrastructure, and potentially triggering a runaway debris chain reaction known as Kessler Syndrome [3], [4].

This thesis project proposes a novel approach to space debris mitigation: utilizing constellations of airborne cubesats for active debris detection and tracking. Cubesats, due to their small size, modularity, and relatively low deployment costs, are well-suited for this task [5], [**swartwout**]. By actively surveying orbital space and providing real-time data on debris location and trajectory, these cubesats can enable evasive maneuvers by operational satellites, significantly reducing the risk of collisions and protecting critical space infrastructure.

1.1 Problem Definition

Space debris poses a significant and escalating threat to operational spacecraft in Low Earth Orbit (LEO) [6, 7]. These hazards originate from various sources, including defunct satellites, spent rocket stages, and even fragments from asteroids [3]. Due to the concentrated nature of space activity within LEO, the following discussion will focus on debris within this orbital region. The detection of space debris, particularly smaller objects, presents a considerable challenge. While ground-based radar systems serve as the primary tool for debris tracking, their effectiveness can be hampered by several factors. These limitations include the technological capabilities of the radar itself, such as sensitivity and resolution, as well as the distance between the radar and the debris object [2]. Despite their relatively low orbital speeds of under 7.8 kilometers per second (roughly 28.000 kilometers per hour), even small debris objects pose a significant threat due to the high absolute velocities encountered in space. Collisions with these objects can inflict catastrophic damage on operational satellites [8]. Therefore, prioritizing the detection of small debris objects is just as crucial, if not more so,

than tracking larger, more easily detectable pieces. The subsequent sections will delve deeper into the specific challenges associated with detecting small space debris objects in LEO.

1.1.1 Scientific Challenges

Detecting space debris, particularly the minuscule objects whizzing through Low Earth Orbit (LEO), presents a formidable challenge. While airborne surveillance systems like LiDAR and optical methods offer a glimmer of hope, their effectiveness is hampered by inherent limitations. Their range is woefully inadequate for vast stretches of space, as acknowledged in[9]. Additionally, the absence of sunlight in the dark regions that do not get sunlight further curtails their utility (refer to[2] for a discussion on the limitations of optical systems). Ground-based radar remains the stalwart of space debris tracking. However, it's not without its Achilles' heel. The very equation governing radar range – the bane of detection specialists – dictates a trade-off between range and resolution (as elaborated in[10]). High-resolution radars excel at unearthing even the tiniest debris objects, but their penetration depth is limited. Conversely, long-range radars struggle to discern smaller objects due to their lower resolution, akin to trying to identify a distant bird through a blurry lens.

The equation further throws another curveball: the exorbitant power demands for extended range. Supplying a single, powerful radar system to cover a vast swathe of LEO becomes an impractical proposition. A potential solution lies in deploying a network of coordinated cube-sets equipped with radars, acting as a distributed and power-frugal detection system (see[11] for a detailed analysis of cubesets for space debris detection). However, the challenges don't end there. Even with improved detection methods, characterizing the material composition and shape of debris objects remains a daunting task. The limited data gleaned from current methods makes it akin to piecing together a puzzle with missing pieces.

1.1.2 Technical Challenges

Beyond the previously discussed physical and logistical constraints, this project encountered a unique set of technical challenges. The most significant obstacle was the scarcity of publicly available data. While radar, LiDAR, and optical datasets exist, they are primarily geared towards the automotive driving industry[12, 13]. These datasets operate at significantly smaller ranges and prioritize the classification of larger objects like cars and pedestrians, a stark contrast to the tiny debris objects we're interested in, where size differences actually aid classification. Additionally, most driving data exists in two dimensions (2D), whereas space debris detection necessitates either one-dimensional (1D) or three-dimensional (3D) data. Although claims of 3D radar datasets from prestigious institutions like ESA and NASA exist, this data remains frustratingly out of public reach[2, 3].

Another hurdle involves the sheer complexity of radar-based computations. Processing and handling radar data is computationally demanding, often exceeding the capabilities of local systems. Even with robust computational resources, data simplification might be necessary to expedite analysis and reduce computational burden. Furthermore, creating simulations that accurately replicate the space debris detection scenario requires meticulous

modeling expertise, which can only be acquired through collaboration with radar and space professionals.

1.2 Motivation

The escalating threat posed by space debris demands innovative solutions. This thesis aims not only to address the existing problem but also to pave the way for future advancements. While countless approaches exist for solving debris detection challenges, this work proposes a novel method that leverages airborne radars and deep learning to unravel the characteristics of space debris. The remarkable progress of deep learning in recent years has opened doors to exciting possibilities. In certain domains, deep learning models have achieved performance rivaling human experts (e.g.,[14]). Inspired by this potential, a key aspect of this thesis will involve evaluating the efficacy of deep learning methods for either bypassing traditional signal processing techniques or recognizing object characteristics within the space debris context.

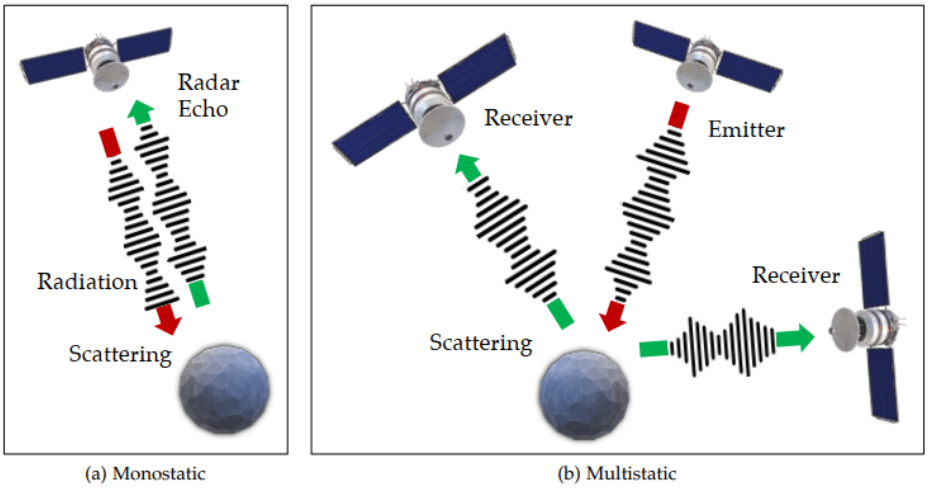


Figure 1.1: Difference between mono-static and multi-static radars.

The scarcity of publicly available datasets for space debris detection presents a significant challenge. To address this, a central objective of this thesis is the development of a realistic data generation pipeline 1.2. This pipeline will be instrumental in training and evaluating the deep learning models employed in this research. Investigating the potential of deep learning approaches to replace or augment conventional methods with the objective of achieving superior performance in space debris detection represents the second major goal of this thesis.

For reasons of efficiency and feasibility, the project focuses on mono-static and airborne radar systems. Difference between mono-static and multi-static radars can be seen in 1.1 .This approach offers a two-fold advantage. Firstly, it prioritizes cost-effectiveness. Instead of relying on a single, highly complex setup that is only accessible to a limited number of research institutions, the proposed method emphasizes the creation of multiple, simpler

devices. The cubesets that can be send to the LEO are small and considering they can be attached with other payloads makes it more convenient than building a super strong radar on ground. Also, the detection setup can be built on an existing other purposed device. Secondly, within the scope of this project, the mono-static structure simplifies radar operation, reducing the level of expertise required. This is particularly beneficial considering the project's timeframe.

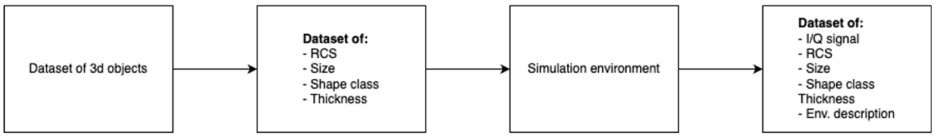


Figure 1.2: Steps of data generation pipeline.

1.3 Radar Principles

Radar, an abbreviation for radio detection and ranging, is a sophisticated detection device that leverages radio waves and their responses. Since it utilizes electromagnetic radio waves, it operates independently of visible light, making it exceptionally valuable in conditions where visible light is unavailable. For instance, radar can function effectively in dark and foggy environments and can even provide vision behind certain objects.

The working principle of radar involves several steps. Initially, the radar generates a signal and directs it outward. Various parameters influence this generation process, which in turn defines the radar's characteristics and type. The generated wave travels at the speed of light until it encounters an object. Upon hitting the object, the wave is reflected in a specific manner, carrying information about the object. The radar's receiver picks up this returning signal and converts it into digital readings using an analog-to-digital converter. Through this cycle, basic information can be derived. By calculating the travel time and dividing it by two, then multiplying by the speed of light, the distance to the object can be determined. Additionally, by employing the Doppler effect, shifts in the frequency domain of the echo can be observed using fast Fourier transformations, which provide the relative speed of the object.

The benefits of radar are evident when considering its resilience to noisy conditions. Unlike light-based visual systems, radars can operate effectively in adverse weather conditions, making them particularly suitable for air and maritime surveillance. Especially, the less effection from the space noise sourced from other objects or space radiation comes in handy in this scenario. Their long-range capability is another advantage, facilitating more effective exploration missions and enabling the analysis of distant objects in space. Furthermore, multiple output multiple input radars can track objects by leveraging Doppler shifts. Given the initial coordinates of an object, its trajectory and speed can be deduced from the Doppler shift, allowing for precise predictions of the object's future position without human intervention. However, the specific information obtained and its accuracy depend on the radar's design and working principle. The two most common radar categories are Mono-static Radar and

Bi-static Radar. Mono-static radars use the same antenna for both emitting and receiving radio waves, whereas Bi-static radars employ separate antennas for transmission and reception. Also within those systems, by using various modulations and transmitter-receiver setups obtaining different outputs is possible, such setups and capabilities can be seen at 1.3.

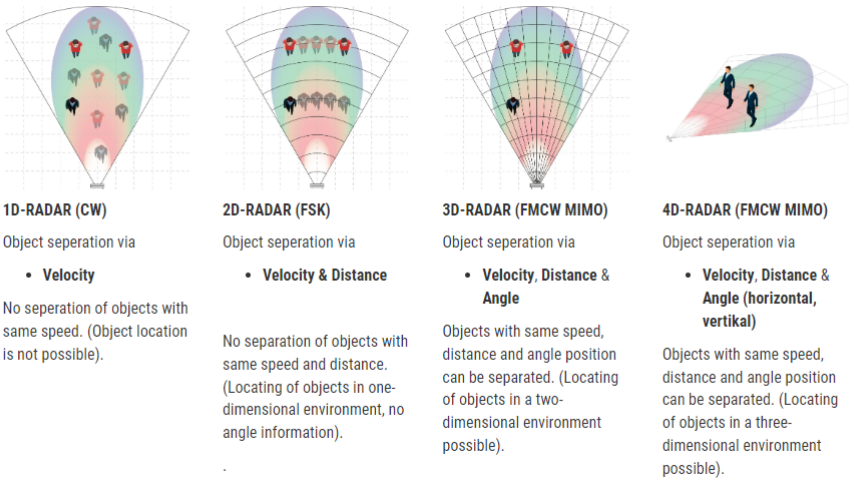


Figure 1.3: Different types of radars and which information can be obtained from them with conventional methods.

Mono-static radars are predominantly favored for their simple design and cost efficiency. Their work cycle differs slightly from the general radar principle. Although the overall concept is similar, Mono-static radars have distinct transmit and listen cycles. This adaptation is necessary because a single antenna cannot meaningfully transmit and receive signals simultaneously. Typically, the received signal is much weaker and undergoes several preprocessing steps. It is first amplified and then filtered to remove noise, after which the relevant information is extracted from the processed signal. Common applications of Mono-static radars include air traffic control, weather monitoring, speed enforcement by police, and autonomous driving systems in the automotive industry.

$$P_t = \frac{P_t G_t G_r \lambda \sigma}{(4\pi)^3 R_t^2 R_r^2 L} \tag{1.1}$$

The radar power equation 1.1 provided elucidates several key concepts. P_t represents the maximum power level of the radar, measured in watts. G_t, the antenna gain for transmission, indicates the efficiency with which the antenna transmits energy in a specific direction. A higher gain signifies more concentrated and less scattered energy. Similarly, G_r is the gain for receiving, following the same principle. Both gains are unitless. The wavelength, λ, is measured in meters and is essential in defining characteristics such as range. The radar cross-section, σ, measures how effectively the target reflects the radar signal. This parameter depends on various attributes of the object, including material, size, and shape, and is measured in square meters. Within the project, the radar cross-section plays a crucial role as it

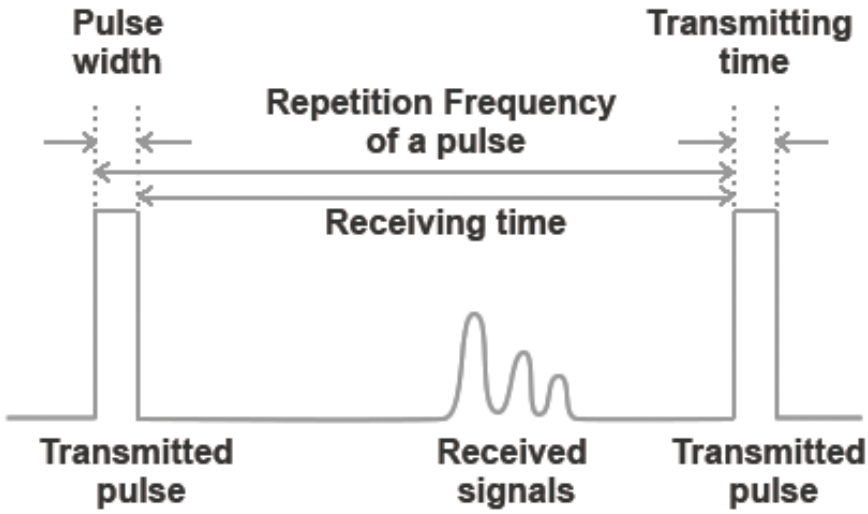


Figure 1.4: Example work cycle of a pulsed radar: First a pulse is generated according to pulse width. Then pulse hits the obstacle and returns back. After the pulse generation, the antenna switches to the listening/receiving time to read received signals.

provides the most relevant information about the object. The distance to the object, R_t, is also represented in meters. In mono-static radars, R_t and R_r (the distance for reception) are the same. L accounts for overall system losses that must be considered in real-world applications.

Briefly examining some of the relationships in the equation, a larger radar cross-section (σ) implies easier detectability due to a larger reflective surface or superior reflective material. The smoothness of the surface, in addition to the material, is also significant. Non-smooth surfaces can reflect the wave in various directions, causing natural noise or unrealistic responses which is an effect intentionally utilized in stealth aircraft design. The equation illustrates that achieving a greater range necessitates higher transmission power. Similarly, shorter wavelengths (λ) result in smaller signal loss but require more power. Understanding these parameters and their relationships is fundamental for optimizing radar performance and applications in various fields, from aviation to autonomous driving. This whole cycle can be observed in the Figure 1.4

1.4 Radar Setup

Designing a radar setup from scratch is beyond the scope of this thesis. Therefore, an existing radar setup specifically designed for space-borne orbital debris detection is adopted. This radar is optimized for detecting objects smaller than 10 centimeters, making it ideal for our purposes [15]. A frequency of 94 gigahertz is selected to achieve high antenna gain. This operating frequency ensures that objects within the 1 to 10 centimeter range remain in the

optical reflection region, thereby maximizing the Radar Cross Section (RCS) return.

The choice of pulse repetition interval (PRI) and pulse width is critical. A lower PRI improves coverage by allowing more pulses during the illuminated time. However, it must also be sufficiently long to define the maximum range effectively. Concurrently, the pulse width should be narrow enough to maximize coverage. Based on these considerations, a pulse width between 20 to 30 microseconds and a PRI between 80 to 100 microseconds are selected. The referenced article provides a detailed explanation of the radar setup, including all relevant variables. Given the simplified nature of the simulation setup in this project, not all features of the radar setup are implemented. Nevertheless, this setup can detect the desired objects within a 15-kilometer range. The chosen parameters are listed in Table 1.1.

Table 1.1: Radar parameters used in the simulation setup obtained from [15].

Parameter	Value
Frequency	94e9 Hz
Pulse Bandwidth	50e6 Hz
Pulse Width	30e-6 Seconds
Number of Pulses	300
Pulse Repetition Frequency	1/(80e-6) Hz
Antenna Beamwidth	0.25 degrees

2 Literature Review

Utilizing airborne radars for detecting space debris is still an evolving field. Despite its nascent stage, the domain is rich with diverse methodologies aimed at detecting space debris and classifying objects under various conditions. Scientists have proposed numerous approaches to space debris detection, acknowledging the significant costs and technical challenges involved. On the other hand, the application of radar technology in the automotive industry is well-established, primarily due to its numerous advantages over alternative sensing methods.

Furthermore, when considering the task as a form of signal processing enhanced by deep learning, the literature expands significantly. While each specific task may appear distinct, many methodologies can be adapted and applied to the domain of space debris detection. By altering the approach and leveraging advances in deep learning, it is feasible to repurpose existing research and techniques from related fields to enhance space debris detection capabilities.

This section will explore relevant scientific articles from various perspectives. Initially, it will review overall methods for space debris detection, both with and without the use of radars. Subsequently, it will delve into the performance of radar systems in the automotive industry, examining their effectiveness with different types of input data and drawing parallels to the space debris detection domain.

2.1 Radar Space Debris Detection

In "Comparison of EISCAT Radar Data on Space Debris with Model Prediction by the MASTER Model of ESA"[16], the threat posed by objects sized 1 to 10 centimeters was identified at an early stage. For detection, data was obtained from observations using the ESA MASTER/PROOF'99 model. This model provides a conventional approach to estimating the likelihood of a spacecraft colliding with space debris.

Some articles, such as[17], focus on the Kessler Syndrome and classify possible solution approaches. These approaches are categorized as "mitigation (to limit the creation of more debris), remediation (to remove debris from orbit), and space situational awareness (to prevent operational satellite collision)". For mitigation, it is noted that authorized institutes provide guidelines, but their voluntary nature renders them ineffective. For active debris removal (ADR), several valuable approaches are proposed, but technical challenges and funding issues have so far made them inefficient solutions. Regarding detection, ground-based radars can provide warnings to satellite owners 72 hours in advance[18]. However, this information, managed by the U.S. military, is not publicly accessible. Additionally, older ground-based

systems can only detect objects larger than 10 centimeters. Among possible solutions, there are also airborne optical systems, such as the IDEA satellite from Kyushu University.

Some research has explored the possibility of detecting space debris using cosmic-ray physics. In one project, researchers employed a telescope called Mini-EUSO, which can observe light reflected from space debris during the day and laser reflections at night[19]. Around 2019, machine learning approaches began to be applied in this domain. By using photometric signatures, defined as "time-varying wavelength-dependent apparent magnitude of energy (e.g., photons) scattered off an object along the line-of-sight to an observer"[20], researchers tested decision trees, ensemble classifiers, linear discriminants, naive Bayes, k-nearest neighbors, and support vector machines. For the preprocessed dataset of light curves (photometric signatures), SVM performed best with an overall accuracy of 98%, demonstrating the potential of machine learning methods in space debris detection.

A review conducted at the University of Cagliari analyzed various ground-based radar systems and compared their performance in detecting space debris[21]. The study evaluated the capabilities of radars from various space agencies. The radar with the maximum range is Cobra Dane in Shemya Island (USA), with a minimum detection size of 5 centimeters. The smallest size detection is achieved by the TIRA-Effelsberg radar in Wachtberg, which can detect objects as small as 0.07 centimeters in diameter at a range of 1000 km. Around the same time, ATLAS was introduced in Portugal, featuring a more sophisticated, fully digital signal processing method [22].

Radar signal processing is computationally intensive. Ground-based debris detection radars, being even more sophisticated, further increase processing complexity. To overcome difficulties in accessing data due to institutional constraints or sensitivities, simulations can be introduced. In 2021, Martin et al. developed a simulation environment for testing different radar setups and generating their signal returns[23]. They enhanced the simulation with Nvidia CUDA support to utilize GPUs for faster processing. The goal of the article aligns closely with the latter part of this thesis, which focuses on a complex data generation pipeline. However, the simulation described in the article is specifically for ground-based radars.

Another approach, which does not involve machine learning, uses the canonical polyadic tensor decomposition method. By leveraging Terahertz channels with limited scattering, canonical polyadic tensor decomposition is framed as an optimization problem and solved using the alternating least squares algorithm. The setup included 64 uniform linear array antennas in an airborne system. Additionally, the article proposes a straightforward setup for LEO satellite transmission, allowing the received signal to be converted into a canonical polyadic tensor[24].

2.2 Automotive Industry Solutions

Radars have diverse applications across various fields, but one area where they are becoming increasingly prevalent is in the automotive industry. With the advent of advanced driver assistance systems (ADAS) and autonomous vehicles, radars are utilized to monitor changes in the external environment and are integrated into land vehicles. These observation modules

enable self-driving autonomous vehicles by providing robust detection capabilities. Radar technologies have outperformed other observation systems due to their insensitivity to adverse weather conditions and their ability to detect Doppler shifts. Furthermore, the integration of machine learning and deep learning approaches with radar technology has enhanced the functionality and reliability of these systems.

There are multiple approaches to using radars in land vehicles. Some methods rely on processed data derived from raw radar signals, building recognition systems on top of this pre-processed information. Others use the raw radar output directly, embedding all processing steps within the machine learning models. The following sections will explore these two approaches in detail, highlighting their methodologies and respective advantages.

2.2.1 Raw Signal Solutions

DeepReflecs emerged from the need to enhance the performance of radar-based systems while maintaining computational efficiency, particularly when integrated with other methodologies such as cameras. Traditional methods that use features built upon spectrograms often achieve high performance but are computationally expensive. To address this, DeepReflecs introduced a novel system combining two core ideas. The first step involves extracting features from radar reflections, such as range, radial velocity, radar cross-section (RCS), and azimuth angle. This information is then fed into a neural network designed to classify objects into categories like pedestrians, cars, and cyclists. This two-step approach enables the system to remain computationally efficient while maintaining accuracy[25].

Another study addressed the challenge of poor angular resolution in radar systems and proposed an innovative solution. Traditional methods relying on range-azimuth maps are critiqued for their high computational cost. Instead, high-definition (HD) radars can be utilized for the segmentation of free spaces. The proposed solution, FFTRadNet, incorporates a unique MIMO pre-encoder with specialized convolutions, an FPN encoder that uses residual connections to convert range-angle representation to range-azimuth representation, and multitask heads for segmentation and detection. This model performs binary segmentation to identify free spaces using raw radar data input[26]. Building on this approach, T-FFTRadNet was introduced, which is based on the Transformer architecture and includes a novel layer called Fourier-Net. This layer allows the model to learn how to perform transformations, moving beyond the conventional Fast Fourier Transformation[27].

Another approach involves using radio frequency images to generate heatmaps. Y. Dalbah et al. developed a network that integrates a Transformer for feature extraction and RODNet for segmentation tasks. This model performs three-class recognition and segmentation for 2D radio frequency images, demonstrating significant potential for radar-based object detection and classification[28]. Furthermore, ADCNet presents a novel method that processes raw analog-to-digital converted (ADC) data, incorporating traditional signal processing concepts within a deep learning framework. ADCNet achieved state-of-the-art results, particularly in categories defined as "easy" in the evaluation criteria, highlighting its effectiveness and robustness[29].

Research by Major et al. introduced a technique for radar-based road user classification

using raw radar signal data without intermediate representations. They utilized a neural network trained on raw radar data to directly classify road users, achieving competitive accuracy with significantly lower computational costs[30].

Schumann et al. explored the use of raw radar data for vehicle detection and tracking. They proposed a deep learning architecture that processes raw radar cube data to extract features for vehicle detection, demonstrating improved performance over traditional radar processing methods[31].

Additionally, Kim et al. proposed a novel approach using raw radar signals for human activity recognition. Their method leverages the temporal and spatial information in raw radar signals to classify different human activities, showing promise for applications in autonomous driving and smart environments[32].

2.2.2 Complex Data Solutions

Solutions that utilize radar output in more complex and processed ways have gained significant popularity. These methods often involve conventional transformation approaches and require sophisticated setups. Techniques such as range-Doppler, range-azimuth, and range-azimuth-Doppler segmentation are commonly employed. Decourt et al. introduced a novel approach by incorporating the segmentation task into a timeline using recurrent convolutional networks. This approach leverages temporal information to improve segmentation accuracy in dynamic scenarios[33].

With the rise in popularity of Transformer architectures, their application in radar data processing has also become prominent. TransRSS is a noteworthy example that utilizes Transformers supported by convolutional layers applied to range-azimuth and range-Doppler views of radar scenarios. This hybrid approach enhances the model's ability to capture both spatial and temporal dependencies in the data, leading to improved performance in various radar-based tasks[34].

Taking data sophistication a step further involves utilizing multiple views of range-angle-Doppler to create a data cube. In the "TransRadar" study, a Transformer architecture is employed to address the challenges posed by radar data, such as noise and sparsity. The Transformer model in this study achieved state-of-the-art results in segmentation tasks on the RADIal[26] and CARRADA[35] datasets, demonstrating the effectiveness of this approach[36].

Another innovative method is presented by Kim et al., who utilized a 3D convolutional neural network (CNN) to process range-Doppler-azimuth data cubes. This model effectively captures spatial relationships within the radar data, leading to significant improvements in object detection and classification tasks[37].

Additionally, Li et al. proposed a method that integrates radar and LiDAR data for enhanced object detection. By fusing the complementary strengths of these two sensor modalities, the proposed system achieves higher accuracy in detecting and classifying objects, particularly in challenging environments[38].

Furthermore, the research by Major et al. explored the use of a recurrent neural network (RNN) combined with radar data to perform sequence prediction and classification tasks.

This approach leverages the temporal dynamics of radar signals, enhancing the model's ability to understand and predict the behavior of detected objects[39].

3 Data Generation Pipeline

One of the purposes of the project is to fill the need for annotated radar reading data for multiple classes of small objects. Within the scope of the project, there will be two types of data generation pipelines will be used. Both of the approaches use 3D objects and their radar cross sections. One of them directly utilizes the information within the radar cross section in a simple way. Another brings a more sophisticated and realistic approach for future uses.

3.1 Shapenet

ShapeNet[40] is a comprehensive dataset widely used in computer graphics, computer vision, and robotics research[40]. It offers two main categories: ShapeNetCore and ShapeNetSem[40]. The core dataset boasts a larger quantity of objects, featuring around 51,300 objects across 55 categories. However, it offers fewer classes compared to ShapeNetSem[40]. While containing fewer objects (approximately 12,000), ShapeNetSem provides a wider range of classes (270)[40]. For the purposes of this project, categories will be referred to as classes. Each object within ShapeNet is unique and possesses an identifier within its respective category. Notably, ShapeNetCore allows for subcategories within a class. For instance, the "airplane" class might encompass various subcategories like bombers, fighters, jets, and even fictional spacecraft[41].

Initially, 10 classes were chosen from ShapeNetCore based on their shape relevance to space debris. However, prioritizing scalability for future research led to a shift towards using the 10 most populated classes. The IQ output analysis is made on these 10 classes but later for simplifiying purposes the data amount class number is reduced. Finally, the number of classes was further reduced to 5 or 6 for the calculation and processing stages to enable a focused analysis of distinct characteristics. Some samples to chosen ShapeNet classes can be seen in Figure3.1 . Despite the existence of extensions like ShapeTalk and ShapeGlot aimed at augmenting ShapeNet, the original dataset remains widely utilized[41]. This popularity likely stems from the value of its manually verified, annotated data and its ease of use due to the WordNet hierarchy integration[42].

One point to clarify is that while ShapeNet objects are claimed to be normalized, this is not entirely accurate. To ensure consistency, all objects are rescaled within this project. Additionally, the provided .obj format, which can define multiple meshes and materials, is incompatible with the current processing pipeline. Consequently, the objects are converted to a new format, resulting in the loss of material information.

Figure 3.1: 6 Sample objects from Shapenet.

3.2 Introduction to the Watertight Concept

In the realm of 3D modeling, a watertight object is defined as a closed mesh that is helds properties such as; each edge connects to two adjacent vertices, each surface encloses the volume without holes, and no surface without volume exists. This is essential for accurate physical simulations and real-world applications. 3D objects created in software environments can often have holes or non-manifold structures, which are impossible in the real world and can disrupt calculations based on physical laws. For instance, in a software environment, two planes can overlap perfectly without volume, as they are sets of relative coordinates. However, such a scenario is impractical in real life. Since calculating the radar cross-section (RCS) involves the physical properties of real objects, the objects used in these simulations must also be watertight.

To ensure an object is watertight, three critical conditions must be met. First, all edges must connect exactly two faces, preventing any overlapping and ensuring structural integrity. This condition guarantees that every edge serves as a boundary for exactly two adjoining faces, forming a coherent mesh. Second, there should be no stray vertices or points floating in space, as a dot without volume cannot exist in the real world. Stray vertices can create inconsistencies and errors in simulations, as they represent non-physical anomalies. The final constraint is that each point on a surface must be either inside or outside the surface, with no ambiguity. This ensures that the object is a solid, enclosed entity with no gaps or undefined regions.

The concept of watertightness is crucial in various fields. 3D printing is one of the most prominent applications, as it involves converting artificial objects directly into physical ones. Non-watertight objects cannot be printed because the printer needs a continuous, enclosed surface to create a solid structure. In game engines and simulations, watertight objects are essential for accurate collision detection and lighting calculations. Without watertightness, objects might not interact correctly, leading to visual and functional errors. Similarly, in animation and rendering, only a watertight mesh can produce smooth animations and realistic rendering. Non-watertight objects can cause visual artifacts and inaccuracies in rendered scenes, diminishing the quality of the final output.

Typically, ensuring an object is watertight is a manual process performed by skilled designers using specialized software tools. This process involves meticulous inspection and adjustment of the mesh to meet the watertight criteria. However, manual watertight modeling is not within the scope of this project, nor do I possess the expertise required for it. Moreover, relying on manual methods contradicts the goal of scalability, as half-automated systems are significantly less efficient than fully automated ones. Manually processing a thousand objects could take weeks, if not months, making it impractical for large-scale projects.

To address this challenge, an additional pipeline has been developed to automate the process of making objects watertight and converting them to the required file format. This pipeline uses advanced algorithms to detect and correct non-manifold structures, fill holes, and ensure all edges connect exactly two faces. It also identifies and removes stray vertices, ensuring the mesh is a continuous, enclosed surface. The automated approach significantly reduces the time and effort required to prepare 3D models for simulation and real-world

applications, making it possible to handle large volumes of objects efficiently. This innovation not only enhances the accuracy and reliability of simulations but also enables broader application of 3D modeling techniques in various fields.

3.2.1 Trimesh

Trimesh[43] is a versatile Python library designed for manipulating and creating 3D objects defined as a set of triangles. It is particularly useful for loading meshes into a Python environment for analysis and is specifically suited for working with watertight surfaces and objects, making it highly relevant to the needs of this project. One of Trimesh's strengths lies in its compatibility with various 3D object formats, including:

- STL: Stereo Lithography
- OBJ: Wavefront Object
- PLY: Polygon File Format
- GLTF/GLB: GL Transmission Format
- 3MF: 3D Manufacturing Format
- DXF: AutoCAD Drawing Exchange Format
- SVG: Scalable Vector Graphics

The ease of use and versatility of Trimesh make it a popular choice for scientific computing, which is the focus of this project. Beyond this, Trimesh is also employed in 3D printing and various computer vision projects. While it does not specifically support radar readings for objects, Trimesh does provide sensor simulation capabilities for LiDAR systems. This functionality allows for the creation of LiDAR-required calculations solely with Trimesh, highlighting its broad applicability and utility in different domains.

Trimesh's capabilities extend beyond these primary uses. It can perform union and intersection operations, as well as smoothing or decimation to adjust the quality of the mesh object. The union operation was particularly useful in this project, as the initial .obj objects consisted of multiple meshes that needed to be merged. Trimesh can also check the integrity of meshes and attempt to fix issues, although in practice, a dedicated script using only Trimesh for fixing and converting objects was found to be inadequate. Objects were either not properly fixed or lost their essential characteristics during the process.

Within this project, Trimesh is utilized for two main purposes. The first is the conversion of data formats without compromising the structural integrity of the objects. The second is the scaling of objects to the required size. Since the .stl data format is unitless and defines only relative coordinates and scales, specifying the unit is essential each time .stl files are loaded. During conversion, the unit information is often lost, which can lead to unreliability or misalignment with the project goals. By using Trimesh, each object is scaled according to the specified unit, ensuring accurate representation.

3.2.2 Manifold and ManifoldPlus

Manifold and ManifoldPlus are open-source GitHub projects accompanied by relevant academic articles[44]. These projects aim to obtain manifold surfaces based on input 3D objects. The fundamental approach involves utilizing a uniform grid to store relevant distances from key points of interest within the object. Initially, the objects are normalized into an artificial cube, with the cube's center serving as the origin. Since only the outermost surface is of interest in the process, all sample points within the cube are categorized as either outside the surface, intersecting the surface, or inside the surface. The surfaces between positive cubes and occupied cubes represent the manifold. To preserve the smoothness of the original surface, a non-uniform grid is employed using octree data structures. This enables the grid to have higher resolution around the surface of interest and lower resolution in regions of less interest.

Next, the isosurface is extracted as the faces between cubes intersecting the labeled outside and inside surface points. All boundary cubes are labeled as positive, and if a positive cube is not occupied by a triangle in the original mesh, it is identified as such. The breadth-first search algorithm is then applied to expand the positive cubes. Any remaining cubes not labeled as positive are designated as negative. In cases of ambiguity, such as when a cube face does not align with the original mesh, vertices are split into three or edges into two to resolve the ambiguity. However, even after these steps, cube faces may still not fit into the original mesh. To address this, the projection of cubes onto the original mesh is used to move all extracted vertices to the nearest triangle along a direction parallel to its normals, with a small step. After completing these steps, the object is rendered manifold.

Despite the assertion that the application guarantees the "manifoldation" of any ShapeNet object, there are instances where objects are irreparably broken. To maintain the integrity of the dataset, broken objects are extracted and excluded.

3.2.3 Alternative Sub-optimal Approaches

During the object fixation process, several automatic algorithms from various applications were tested. The primary approach was to start with simpler methods and gradually transition to more complex ones. Unfortunately, all of these attempts proved unsuccessful for fixing complex objects in the ShapeNet dataset. While these algorithms offer functionalities such as hole filling, removing non-manifold edges, and making objects watertight, they were not effective for the specific requirements of this project.

Among the tools tested, Prusa Slicer stood out as the most basic one. Although it lacks modifiable parameters for fixation and its algorithm is not publicly shared, its simplicity makes it a useful tool for visualizing 3D objects[45]. Meshmixer, developed by Autodesk, offers more advanced features and is commonly used in the industry. While it provides specific fixes like hole filling, recalculating normals, merging vertices, and removing duplicate faces[46], it was unable to adequately fix complex ShapeNet objects.

Blender, a comprehensive 3D modeling tool with Python integration, offers a 3D printing add-on that checks the printability of objects. While printable objects must be manifold,

Blender's add-on for making objects printable did not yield satisfactory results for this project. Additionally, Blender provides operations such as "bridge edge loops" and "remesh," but these options, either individually or in combination, failed to address the issues with the ShapeNet objects[47].

Windows 3D Builder, built into the Windows operating system, offers a simple fix option when loading objects. While the algorithm behind it is not shared, it proved to be the closest to a successful approach until the final solution pipeline was developed. In terms of complexity for the user, Prusa Slicer and Windows 3D Builder are the simplest options, requiring only a single button click. Meshmixer offers a slightly more sophisticated approach that allows for some preferences to be set. Blender, while the most complex tool among the options, offers the most flexibility but requires a higher level of expertise from the user.

3.3 Radar Cross Section Calculation

As with many phenomena in physics, the response of an object to radar waves is highly dependent on specific characteristics of the object. The Radar Cross Section (RCS) is a measure that describes the magnitude of the reflected signal when radar waves hit an object. Represented by the symbol σ, the RCS quantifies how detectable an object is by the radar receiver[48]. The unit of RCS is square meters, which indicates the effective surface area that reflects the radar signal back to the receiver. Additionally, RCS can be expressed in decibel square meters (dBsm), a logarithmic conversion of the square meter measurement.

The significance of RCS extends to a wide range of applications, particularly in defense technology. For instance, air defense systems require a high RCS to ensure reliable target detection, while stealth technology in fighter aircraft aims to minimize RCS to evade detection[49, 50]. Advances in material science, which focus on reducing radar reflections, often use RCS as a critical performance metric. Moreover, the environmental impact of high-RCS objects is notable, as they can increase background noise and clutter in radar systems.

In the context of this thesis, RCS is utilized to extract shape-based features of target objects. Although this work primarily focuses on the shape, it is theoretically possible to infer material properties and object size from the RCS[51]. The ability to discern such characteristics from the reflected radar signal underscores the importance of RCS in various scientific and engineering domains.

Several factors can influence the RCS output beyond the inherent properties of the object. Resonance, for example, can amplify the RCS when the frequency of the radar wave matches the natural frequency of the object. Polarization of radar waves also plays a crucial role, as horizontal and vertical polarizations can produce different RCS values[52]. Furthermore, the aspect angle "the angle between the radar and the object" significantly affects the RCS. Consequently, RCS patterns are often represented in three dimensions to account for variations at different angles. It is important to note that a single radar system cannot capture the complete RCS pattern for all angles; instead, it can only observe a subset of the possible angles[53].

$$\sigma = \lim_{r \to \infty} 4\pi r^2 \frac{S_s}{S_i} \tag{3.1}$$

The calculation of RCS depends on several variables, as shown in the fundamental equations 3.1. The distance r from the radar to the object, ideally infinite for far-field calculations, affects the results. S_s represents the wave's power emitted from the radar, while S_i denotes the power density in watts per square meter. Calculating S_i is complex, particularly for irregularly shaped objects, requiring intricate linear algebra to model the illuminated region of the object. Material properties and multiple scattering phenomena add further complexity due to varying reflection rates and multiple scattering paths.

One effective method for calculating S_s is the Method of Moments (MoM), which segments the object into smaller parts[54]. By calculating the reflection from these individual segments and combining the results, a comprehensive value for S_s is obtained. This method considers object geometry, material properties, and wave characteristics[55].

Physical Optics (PO) is another widely used method for RCS calculation. It employs ray tracing algorithms to simulate the radar wave's interaction with the object, treating the wave like a light beam[48]. This approach accurately models reflection, diffraction, and other scattering effects, provided the parameters are appropriately set. Like MoM, PO takes into account the same dependent variables[56].

Additional methods include the Fast Multipole Method (FMM)[57], the Boundary Element Method (BEM)[58], and the Physical Theory of Diffraction (PTD)[59]. BEM, an alternative to MoM, solves surface equations using integral techniques [60]. PTD enhances edge scattering accuracy, while FMM accelerates calculations for large objects[61].

3.3.1 Matlab Approach with Parallel Pool

There are various tools and methods to calculate RCS in a computational environment, with Matlab[62] being one of the most reliable options. The antenna toolbox in Matlab includes a function called `RCS`, which takes various inputs for calculation parameters. The primary input is the object, which can be either a Matlab-defined geometric shape or a platform with an .stl type object. This requirement necessitated the conversion of ShapeNet object data to a compatible format. Frequency is another parameter specified as a scalar number. The method allows for the specification of azimuth and elevation from the viewing angle perspective, and either of these spatial parameters can be provided as an array. This results in an RCS slice, with typical configurations being 0 elevation and [-180,180] azimuth or 90 azimuth and [-90,90] elevation. To obtain a comprehensive RCS matrix, both parameters must vary, with elevation from [-90,90] and azimuth from [-180,180]. Additionally, polarization can be defined as "HH", "VV", "HV", or "VH". The function provides three solvers for RCS calculation: Physical Optics, Method of Moments, and Fast Multipole Method. Of these, only Physical Optics is compatible with GPU usage and is also the fastest for CPU usage in the tested environment.

The azimuth and elevation configuration is crucial for comprehensive RCS analysis. For a full 3D RCS matrix, both elevation and azimuth values need to span their complete ranges.

This thorough analysis is necessary to capture the target's RCS from all possible viewing angles, thereby providing a complete understanding of its reflective properties.

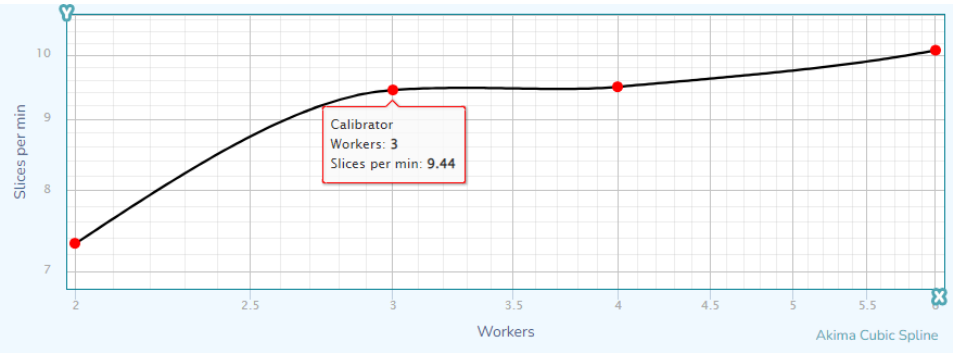


Figure 3.2: Number of calculated Azimuth slices per minute with an increasing number of cores.

Since the method does not directly provide a 3D matrix of size [181,361], it was necessary to modify the approach. Two potential solutions were considered: modifying the `RCS` method or looping through all the slices and merging them. The latter approach was chosen for faster development. However, even this method takes considerable time on local computers since calculating illuminated regions and physical optics for obtaining RCS for any specific angle involves large matrix multiplications. The looping method involves iterating through azimuth and elevation angles, computing the RCS for each pair, and then assembling these into a complete matrix. This approach, while straightforward, is computationally intensive.

The computational intensity is particularly notable when dealing with high-resolution RCS matrices. The increase in resolution directly correlates with the number of calculations required, exponentially increasing the computational load. For example, doubling the resolution in both azimuth and elevation results in four times the number of individual RCS calculations.

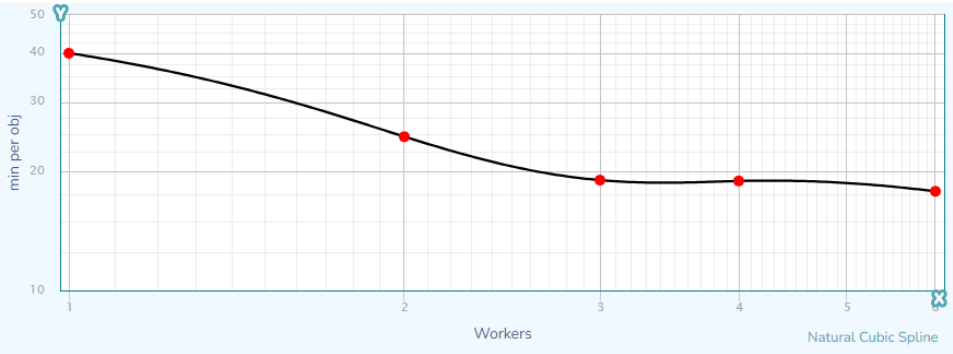


Figure 3.3: Calculation time in minutes per object with increasing number of cores.

To address this issue, a cluster from the chair of scientific computing was requested. The initial cluster, named "HomeOne" did not have Matlab installed so it is solely used for

model training. This issue was resolved by gaining access to another cluster, the "Linux cluster"[63], which already had Matlab installed, obviating the need for further installation. When assigning jobs to Slurm, the job scheduling system differs from running local or terminal scripts. An adaptation to the Slurm system is made, and then code is adapted to it. Typically, 24 cores were utilized for most objects, although memory issues sometimes necessitated reducing the number of cores to 16 to decrease concurrent memory usage.

Utilizing a high-performance computing device was essential for handling the computational demands of this project. The ability to distribute different classes or object calculations across multiple nodes significantly reduced the time required for the generation of the RCS dataset. Also, splitting all angles to a high number of cores reduced the calculation for each object's RCS matrix. As can be seen in figure 3.2 and figure 3.3. This parallel processing capability is particularly beneficial when dealing with large datasets or high-resolution calculations where single-threaded performance would be insufficient.

The Matlab antenna toolbox and its integration with high-performance computing clusters exemplify the power of modern computational tools in addressing complex engineering problems. By leveraging these tools, the project was able to achieve its objectives efficiently and effectively, demonstrating the value of combining advanced software with powerful hardware resources.

3.3.2 Optix RCS

OptixRCS[64] stands out as an open-source application hosted on GitHub, boasting itself as the world's fastest RCS calculator. Developed on NVIDIA's OptiX framework, it employs the method of Shooting and Bouncing Rays (SBR), renowned for its computational efficiency. The developers claim that OptixRCS outperforms its competitor, RaytrAMP, by a significant margin, purportedly achieving speeds fifty times faster.

Installation of OptixRCS requires a few prerequisites. Firstly, users must possess an NVIDIA-compatible GPU, leveraging its parallel processing capabilities for accelerated computations. Additionally, the NVIDIA CUDA toolkit must be installed within the software environment, along with Optix integration. The project itself is structured as a C project, built using CMake, allowing for flexible and efficient compilation.

The functionality of OptixRCS is straightforward yet powerful. It accepts .obj files as input, representing the 3D geometries of objects, and produces the corresponding RCS matrix as a CSV file. Users can specify various parameters, including frequency, azimuth, elevation angles, and polarization, tailoring the calculations to their specific needs.

Despite its promising features, OptixRCS faces certain limitations and challenges. The lack of an official installation guide complicates setup, requiring users to navigate the compilation and execution process independently. Moreover, the project's documentation is sparse, exacerbating difficulties in troubleshooting and resolving bugs. In local environments, encountered issues and unresolved bugs hindered successful execution, hampering further exploration of its capabilities.

An additional barrier to adoption arises from the absence of the NVIDIA OptiX framework in the computing clusters. Without the necessary authorization and infrastructure to install

and configure OptixRCS, exploring its potential becomes unfeasible. Consequently, OptixRCS is currently excluded from consideration as an alternative RCS calculation solution.

While OptixRCS presents itself as a promising tool for rapid and efficient RCS computations, its practical utility is contingent upon overcoming these obstacles and ensuring seamless integration within the computing environment. Efforts to address documentation gaps, resolve technical issues, and establish compatibility with existing infrastructure could enhance its accessibility and appeal to a wider user base.

3.3.3 RaytrAMP: A Powerful RCS Calculation Tool

RaytrAMP[65] stands out as a robust open-source library hosted on GitHub, offering a sophisticated solution for radar cross-section (RCS) calculations. The core of RaytrAMP's functionality lies in its innovative approach, drawing inspiration from the shooting and bouncing rays method, a widely recognized technique in electromagnetic simulations. This method enables RaytrAMP to deliver high-fidelity results, particularly well-suited for scenarios involving high-frequency electromagnetic waves and far-field range calculations.

A notable feature of RaytrAMP is its efficient utilization of GPU resources to accelerate computation. By leveraging advanced GPU techniques such as bounding volume hierarchy (BVH)[66], RaytrAMP optimizes the processing of electromagnetic interactions, resulting in significantly reduced computational times compared to traditional CPU-based methods. This GPU-accelerated approach enhances the scalability and performance of RaytrAMP, making it a powerful tool for researchers and practitioners in radar technology.

One of RaytrAMP's key strengths lies in its compatibility with Mono-static radar systems, a common configuration in radar applications. This compatibility ensures seamless integration with existing radar setups, allowing for streamlined RCS calculations without the need for extensive modifications or adjustments.

In terms of development environment, RaytrAMP supports both C++ and Matlab, providing flexibility for users with different programming preferences. However, one noteworthy aspect is the specific data input requirements of RaytrAMP. It expects 3D object meshes in .unv format, which are then processed and converted into .rba and .obs files. While this adds a layer of complexity to the data generation pipeline, it also enables RaytrAMP to handle diverse input data formats effectively.

In performance benchmarks, RaytrAMP demonstrates superior efficiency compared to traditional RCS calculation methods, such as the FEKO toolset. Despite this advantage, it's essential to consider certain limitations, particularly regarding computational speed under certain conditions. While RaytrAMP excels in scenarios with a small number of directions (looking angles) or low triangle counts in 3D meshes, it may experience performance bottlenecks in more complex simulations.

Overall, RaytrAMP emerges as a formidable contender in the field of radar technology, offering unparalleled efficiency and accuracy in RCS calculations. Its sophisticated algorithms, coupled with GPU acceleration capabilities, make it a valuable asset for researchers, engineers, and professionals seeking advanced solutions for radar system design and analysis.

3.3.4 POfacets

POfacets[67] is a software program designed for solving radar cross-section (RCS) calculations using the Physical Optics method. It operates by computing the response from each triangle of an object and then summing their reflections. This approach allows for the consideration of individual surface features such as material properties and thickness. POfacets offers support for both Mono-static and Bi-static radar configurations, providing users with flexibility in their simulations.

The software can be integrated into the Matlab environment as an external application. It comes in two versions: a classic Matlab version and a more user-friendly Python version developed later. To utilize POfacets in Matlab, it must be installed in the Matlab source directory, after which users can access its interface through a relevant Matlab script. From there, users can define radar setups, import objects, and perform RCS calculations.

Despite its functionality, POfacets has several limitations outlined in its documentation. It struggles with accurately modeling scenarios involving multiple scattering, where radar waves interact with multiple surfaces of an object. Additionally, while the interface provides accessibility, it may not be intuitive for beginners. Furthermore, POfacets is computationally slow compared to other alternatives, with both calculation times and interface responsiveness being notable issues. As a result, it is not selected for this project due to concerns regarding speed and usability.

3.3.5 FEKO Altair

FEKO[68], an acronym for field emission and kinetic electron optics, stands as a comprehensive software solution developed by Altair Engineering company. Its capabilities extend to the analysis and prediction of electromagnetic wave interactions with various objects, making it a versatile tool across different domains. Among its common uses are antenna design and optimization, electromagnetic compatibility assessments, wave propagation analyses, and radar cross-section (RCS) calculations.

Widely embraced for its reliability and diverse solver options, FEKO offers a range of features that contribute to its widespread use in RCS calculations. Notably, it provides solvers such as the Method of Moments and Physical Optics, along with hybrid solvers that combine these methods for optimal accuracy and efficiency. Moreover, FEKO facilitates material distinction, enabling users to simulate a wide array of scenarios with varying material properties. Its capability to account for multiple scattering phenomena further enhances its utility, as many alternatives lack confidence in this aspect. Additionally, FEKO boasts intuitive 3D scenario and object manipulation features, eliminating the need for external applications in many projects.

However, despite its robust features, FEKO has notable drawbacks that limit its widespread adoption. Primarily, its complexity poses a barrier to entry for inexperienced users, necessitating a steep learning curve or reliance on tutorials for proficiency. Moreover, the accuracy of its calculations comes at the cost of extensive computational time, rendering it impractical for personal hardware setups. In the absence of dedicated tools to run FEKO with a prepared

environment, it was deemed unsuitable for the project's requirements, leading to its exclusion from consideration.

3.3.6 Comparison and Performance

This section aims to assess the performance of the aforementioned alternatives from various perspectives, with a focus on scalability which can be counted as the most critical metric for the project's objectives. The pace of calculations is paramount, particularly given the potential scale of the project. For instance, a small difference in processing time, such as between sixty and ninety seconds, can translate into significant differences when dealing with thousands of objects. To provide context, the fastest solution achieved a runtime of approximately five minutes per object. Consequently, even with a thousand objects, the computation time would extend to around 83 hours.

Notably, the comparison of performance is conducted under different hardware conditions, with priority given to the most feasible option for the project's execution environment. Among the alternatives, POfacets and Altair FEKO exhibited the slowest performance. POfacets encountered frequent crashes and required over 5 hours for calculations, while Altair FEKO also exceeded the 5-hour mark for a single object, rendering both options impractical.

OptixRCS followed closely as the second slowest, facing challenges such as GPU memory issues and a lack of documentation, which complicated debugging efforts. Despite these hurdles, the latest stable version achieved a runtime of approximately 3 hours per object.

In contrast, the two fastest options were Matlab and RaytrAMP. RaytrAMP demonstrated a processing time of 15 minutes per object, outperforming the base Matlab implementation, which required around 20 minutes. Notably, RaytrAMP lacks the ability to be parallelized in the cluster environment. Conversely, Matlab's RCS calculation, when parallelized with a parallel pool, yielded the fastest runtime of approximately 5.5 minutes per object, making it the optimal choice among the alternatives.

Table 3.1: Comparison of used methods for RCS calculation.

Method	Advantages	Disadvantages	Runtime
RaytrAMP	Faster than base Matlab	.stl to .unv conversion	15 minutes
OptixRCS	Can already use .obj	GPU memory issues	More than 3 hours
Matlab	Reliable and Fast	Totally Manual	20 mins default, 5.5 parfor
Altair FEKO	Industry standart	Slower than alternatives	More than 3 hours
POFacets	Provided Interface	Crashes Frequently	More than 3 hours

Beyond runtime considerations, each method offers unique advantages and disadvantages. RaytrAMP, for instance, provides a rapid estimation of RCS and can discern differences in distinct datasets. OptixRCS presents a promising approach but requires thorough debugging and documentation to realize its full potential. POfacets may appeal to non-coders with its user-friendly interface, despite its performance limitations. Altair FEKO stands as an

industry-standard solution renowned for reliability and professionalism, albeit with a steep learning curve.

3.4 Analysis of RCS output as slices and 3D

After establishing the pipeline for calculating RCS for a large number of objects, an initial dataset comprising 1016 objects was processed. This dataset consisted of 100 objects from 10 different classes, with the rocket class comprising 16 objects for experimental purposes. However, during the calculation process, some data loss occurred due to a calculation resulting in a division by infinity, resulting in all values in the matrix becoming NaN values. After cleaning up the matrices, 964 samples remained. The entire calculation process took 39 hours and 14 minutes in the Linux cluster, according to Slurm, which is 4 times faster than any closest alternative option.

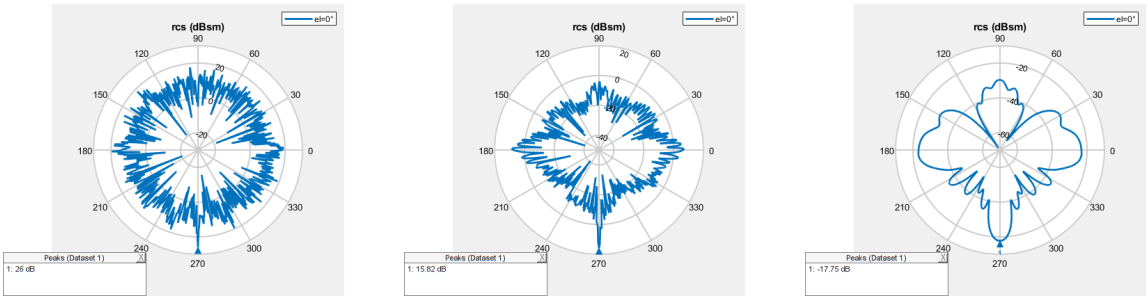
Upon generating the dataset from relevant radar readings, most of the models produced nonsensical results. They either failed to learn the patterns during training or, if they did, they could not generalize. This led to the realization that the dataset's outcome appeared to be random, at least for the models that could be trained under these circumstances. To confirm this hypothesis, several scripts, tools, and analyses were conducted.

The initial comparison and analysis involved observing spherical patterns in the RCS matrix. A special method for plotting the RCS matrix as a sphere, sourced from Matlab code, was adapted to accommodate the dataset's format. Many samples from various classes were then plotted. While some patterns were distinguishable, others, such as those from cars and planes, appeared quite similar. Mathematically inspecting these patterns revealed similarities, with the rocket displaying higher signal magnitude. However, the lack of expertise in radar systems raised doubts about the dataset's accuracy, especially as Matlab claimed its RCS calculation to be insensitive to the distance to the object.

Further inspection involved a Python script to obtain all elements in a class, averaging their RCS matrices, and plotting them for analysis. Although a clear pattern was expected due to the consistent object orientation, only vague differences were observed for some classes. The round plot (figure 3.9) of RCS matrixes among different classes were supporting this hypothesis. The interference of working models in these classes hinted at the dataset's information being excessively noisy, prompting additional studies on RCS outcomes.

In the Matlab platform environment for calculating the RCS, only a few parameters can be modified. Frequency is already fixed due to radar parameters defined for the specific task. Polarization is another important parameter, known to influence signal readings. Therefore, experiments were conducted using Vertical-Vertical and Horizontal-Horizontal polarization. However, these experiments did not yield the expected outcomes. Subsequently, an observation of the spherical presentation of the RCS matrix led to the hypothesis that the problem might be caused by low resolution.

Increasing the resolution comes with a significant increase in computational complexity. For instance, one-degree resolution results in 6480 different angles (360 azimuth angles and 180 elevation angles). A tenfold increase in resolution translates to 648,000 different angles.



(a) Unit in meters (b) Unit in feet (c) Unit in inches

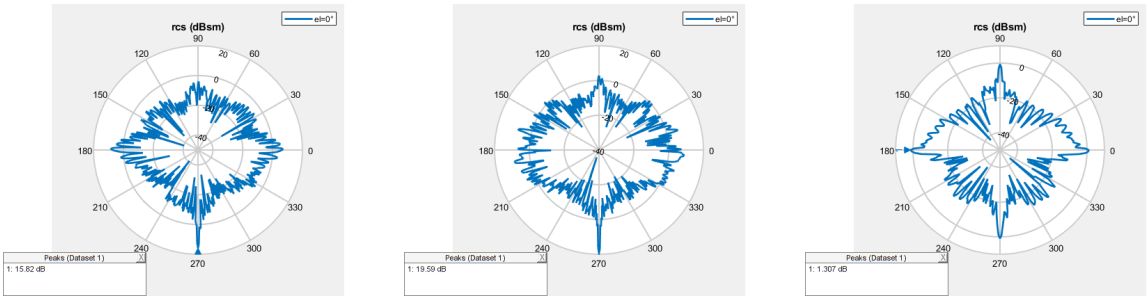
Figure 3.4: Cars RCS slice with 0 elevation.

Even the smallest increase, with half a degree resolution, requires four times the computation. Therefore, trials on resolution were conducted for only one slice of the angles. Specifically, elevation was kept constant at 0 degrees while azimuth was varied from 0 to 360. Upon comparison, it was observed that the two results were exactly the same (check figure 3.7), disproving the hypothesis that the problem or noise stemmed from low resolution.

This elimination draws attention to object-related parameters. Despite applying manifoldization operations, the smooth surfaces in the original objects lost their smoothness. Even with a much higher sampling rate for the manifold algorithm, this issue persisted. In the last trial, which involved 100,000 samples resulting in approximately 600,000 triangles in the mesh, only a finer rough surface was achieved. The RCS matrix still exhibited a jumping behavior, contrary to the expectation that finer rough surfaces would result in smaller jumps.

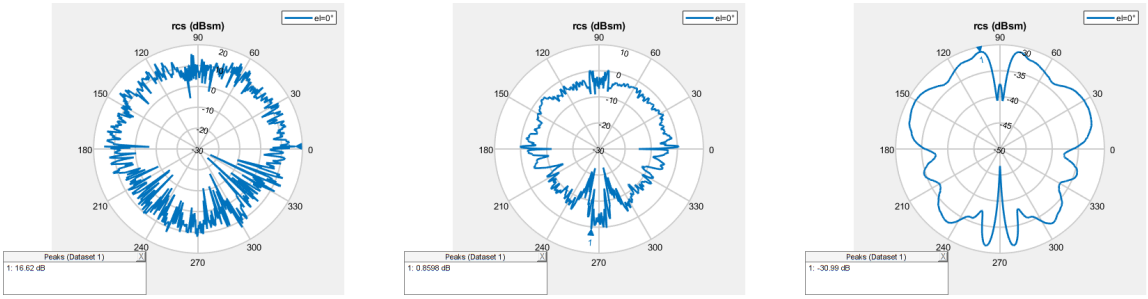
Although the Matlab environment does not inherently include a parameter for the distance to the platform object, the object's size remains a modifiable parameter. In the last trials, an inspection of the same object with different unit scales was conducted as can be seen in figures 3.4 and 3.6. Initially represented in meter units, the slice analysis produced the same output as before. However, when the unit was changed to feet, noticeable changes occurred. A distinct pattern emerged from the slice plots. To verify the consistency between classes, different objects underwent the same process and were examined. Similar classes began to exhibit similar slice characteristics. For instance, cars commonly displayed a low ground in the top left and right-looking angles, while planes exhibited rich signal readings in their looking angles. Additionally, planes received lower values when viewed from the front angle. To further confirm the observations, the unit was changed to inches. With an even smaller metric, most of the noisy jumps were eliminated, leaving behind a smooth pattern.

With guidance from the supervisor, measurements were conducted on all the ShapeNet objects. During this process, another problem became apparent. Despite the claim that all objects were normalized, it was discovered that this assertion was either untrue from the outset or that consistency was lost during the conversion to watertight. To address this issue, a Python script was developed with the assistance of the Trimesh library. This script was utilized to check the size of all objects and rescale them to a standardized measurement of 5 centimeters.



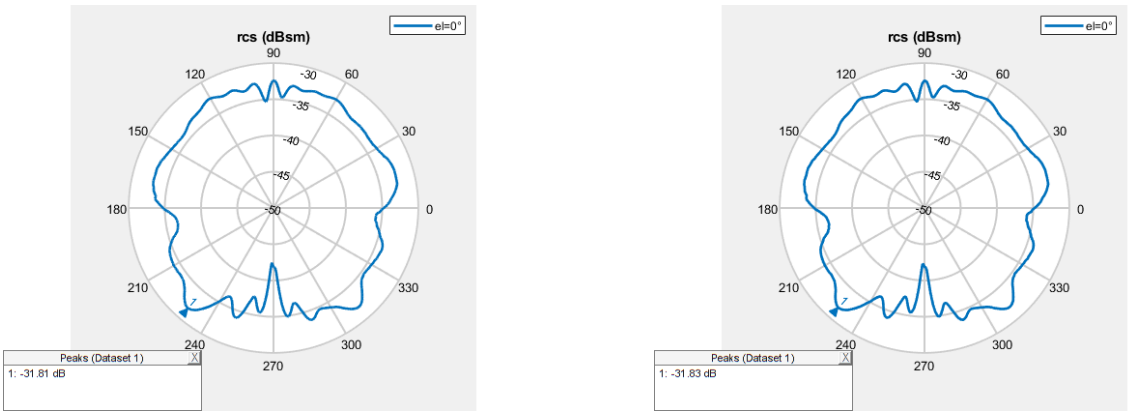
(a) Unit in meters (b) Unit in feet (c) Unit in inches

Figure 3.5: Different car objects with 0 elevation and 0-360 azimuth, unit as inches.



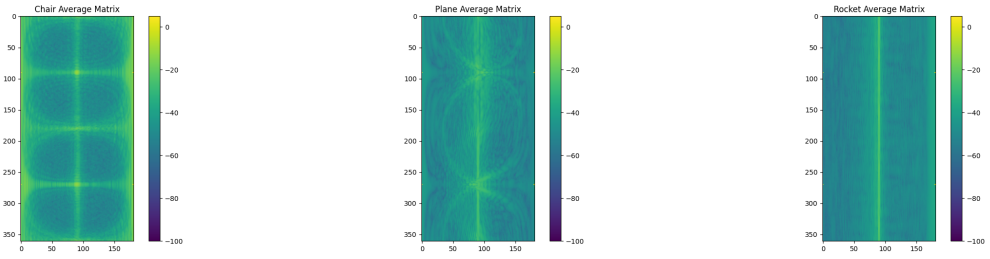
(a) Unit in meters (b) Unit in feet (c) Unit in inches

Figure 3.6: Planes RCS slice with 0 elevation.



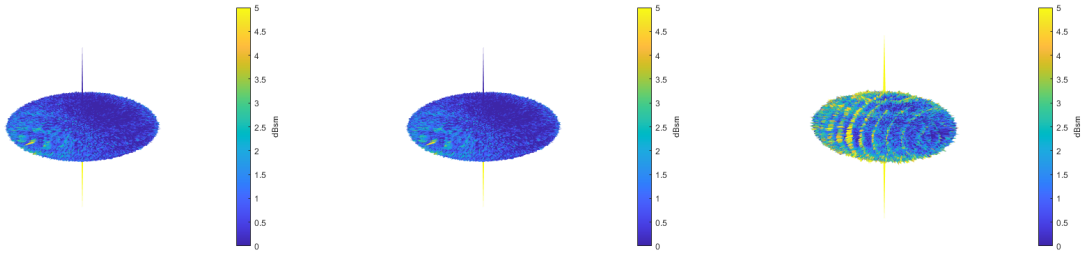
(a) 1 degree resolution (b) 0.5 degree resolution

Figure 3.7: Resolution comparison for Plane.



(a) Chair average matrix (b) Plane average matrix (c) Rocket average matrix

Figure 3.8: Average RCS Matrixes.

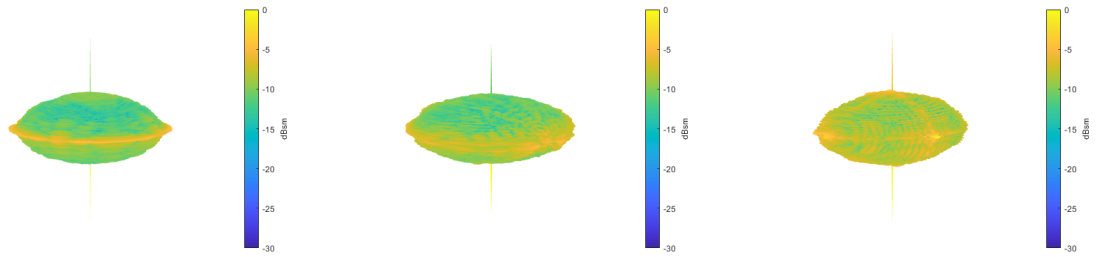


(a) Car sample RCS (b) Plane sample RCS (c) Rocket sample RCS

Figure 3.9: Sample round RCS matrixes.

All of the RCS matrices were recalculated using the newly scaled images, ensuring that they were standardized to a size of 5 centimeters. This standardization ensured that the longest diagonal in each object could not exceed 5 centimeters. Following the recalculation of these RCS matrices, previous trials were conducted to observe the characteristic differences between classes. The initial trial involved examining individual objects in the round plot of RCS(figure 3.10). The patterns observed were much clearer: samples from rockets exhibited a stronger signal around the equator, planes displayed a unique cross-like following pattern due to their large surface wings, and cars showed clear peaks at points that were perpendicular to the object (figure 3.5).

To generalize the distinction among classes, the average RCS matrix for each class was computed by averaging all the objects within that class. These average matrices were then plotted as color images using the same class-average method in Python. The results clearly showed that the differences between classes were easily observable and distinguishable. As depicted in Figure 3.8, chairs exhibited a pattern that divided the matrix into square-like features. The elliptical shape caused by the wings of planes was clearly observable, distinguishing them from other classes. Additionally, rockets displayed a flat line pattern, characteristic of their consistent cylinder-like structure.



(a) Rocket (b) Plane (c) Car

Figure 3.10: Round RCS plot with new units.

3.5 Simulating Radar Return

The simulation of radar return involves more than just calculating the Radar Cross Section (RCS) of an object. While RCS determines the pattern of reflection, the real signal return is a multifaceted process that requires consideration of various additional parameters. These include factors such as signal propagation and the possibility of interceptions occurring in Low Earth Orbit (LEO). Moving forward, two distinct data generation strategies will be employed to simulate radar returns.

The first approach, inspired a prior reseach done internally in Actlabs, is a simpler yet effective method. It focuses on a subset of variables to assess the capability of deep learning models in object recognition tasks. This approach offers an efficient means of generating data for training and validation purposes.

The second strategy involves a more comprehensive simulation process, incorporating a broader range of variables and scenarios. While this approach is more complex and resource-intensive, it provides a more detailed representation of real-world radar returns. By exploring both approaches, we aim to gain insights into the performance of deep learning models across different simulation settings and scenarios.

3.5.1 Slice Approach

The slice approach in radar return simulation introduces a novel concept distinct from traditional slicing in RCS matrices, which typically involves constant elevation and considers azimuth as a whole. Here, the concept is derived from potential observation points on the object, which may follow linear or nonlinear trajectories with some rotation speed.

In this approach, the distance to the object becomes irrelevant, and the rotation speed is represented solely in degree units. Additionally, the object itself remains stationary. Given that RCS reflects the magnitude of reflection at a specific point, and assuming other parameters in the radar equation are fixed, it becomes feasible to replicate a returning signal for a specific condition.

The methodology involves selecting a random point on the RCS matrix of an object,

followed by choosing a constant or exponential rotation value. At each timestep, the angle of observation transitions to the next point in a random direction, with the rotation value dictating the magnitude of this transition. Here, each element in the matrix can be considered as cells.

Since rotation is defined in degrees, there is no need for unit conversion. However, there is a possibility that the next landing point may fall between cells or not precisely on the cell centers. To address this, the bilinear interpolation approach is employed. While more complex interpolation solutions could be adapted, it's essential to note that interpolation represents the most time-consuming aspect of data generation in this scenario. Therefore, implementing a more complex interpolation algorithm may significantly increase complexity.

3.5.2 phased.BackscatterRadarTarget Function for Swerling Models

Implementation of the slice approach is made in the Matlab environment as a whole. The first thing that needs to change is to adapt to the Slurm job assignment environment. Then rotation possibilities are limited to 1 and 0.5 degrees. In the original version, it goes as low as 0.05. The restriction is because the sample objects in the original GitHub page have much more resolution regarding the RCS matrix. In this case, it would be like looking at a low-pixel image with a magnifier.

The function takes two parameters: first, the path of the class that consists of the relevant objects with IDs, and second, the object's name for labeling purposes. So that the data labels can be different from the semantic name of the class for possible future purposes. After taking the input parameters into account, frequency, and speed of light are defined for the requirements phased.BackscatterRadarTarget function. Also, the number of samples that wanted to be generated from one object and how many timesteps each sample would be defined as well.

The RCS matrix is loaded into the environment for each object under the given folder. Relevant azimuth and elevation arrays are defined. Also, the obtained RCS matrix is transposed to meet the phased convention.BackscatterRadarTarget function. It returns a system object that consists of scattering values for given angles. The function doesn't support polarization with RCS, but since RCS calculation of complex objects already considers polarization, it is represented here. This is another deviation from the original GitHub repository. The method takes a Swerling fluctuation model as well. In the default, no Swerling model is defined. For this case, the Swerling 2 model is chosen for its suitability for pulsed radar systems. Every value at each time-step is obtained from the base method of the "simulateTrajectoryPaths" method. Finally, it is possible to add the Gaussian noise that is wanted to the model in the base method. But it has not been added to this project. The logic behind the approach is relevant to the rough surfaces that are supposed to be smooth. They are adding a noise in their nature, and it is guaranteed no perfect smooth surface exists.

3.5.3 Prior Actlabs Development

Further details about the project idea and code template for the slicing approach will be discussed in this subsection. The prior research in Actlabs that I used as inspiration was conducted by Oscar Breiner. His work was focused on recognising simple shapes such as cones, cylinders, spheres and circular planes. The work also explored differences in performance for several DL methods such as MLP, LSTM, ResNet and Transformers as well as robustness. His work mostly focused on predicting geometric shapes like cones, cylinders, spheres, and circular plates in a similar environment.

The first approach of data generation in this thesis is inspired by his work and satisfies the curiosity about whether it is showing a similar behavior with complex objects as well. His project also covers a robustness analysis for various deep learning models that are not taken into account for this project. Even though his work is a comprehensive study, only the alternative data generation approach is used in this project. Hence, his work is more focused on deep learning performance, and this project is more focused on data generation in a more realistic way. As foreshadowing a further project that merges both into account can be made to test if any solutions are promising for more realistic scenarios.

4 Deep Learning Models

To see the results of the first data generation approach and use it as proof for further complex data generation pipeline potential, a set of deep learning models is trained. Since conventional signal processing relies on statistical approaches, it cannot develop itself further with the data fed. Even good-performing approaches take external ground-breaking ideas or perspectives to take it one step further.

Deep Learning can leverage the data and use it to surpass the static nature of the conventional signal processing processes so that it can perform faster and more reliably in low earth orbit where time and accuracy are most valuable. To mention more specific details, conventional radar processing approaches apply static mathematical and statistical steps that treat all of the data equally. On the other hand, fast-response deep learning models can learn where to look and bring a more sophisticated inspection. In this project, only radar readings are fed to the network architectures as data for simplicity reasons. However, it is possible to extract more information due to the nature of the real radars. In addition to readings, all the information can be fed into the data for further predictions and faster convergence.

4.1 Used Architectures and Training Environment

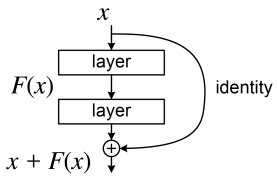


Figure 4.1: Logic behind ResNet architecture.

In the pursuit of an effective solution, various Deep Learning approaches were selected according to their popularity, acknowledging the inherent complexity even in simplified models. The diverse conceptual frameworks underpinning different models can significantly influence their success or failure. Therefore, a range of models was systematically evaluated, ordered from simple to complex.

The Multilayer Perceptron (MLP) is the initial and simplest approach. Despite its simplicity, the MLP provides a foundational understanding of the problem domain and offers a baseline for performance comparison[69]. MLP is the most basic feed-forward neural network. It consists of multiple layers; among those layers, there are multiple nodes. There are three types of layers: Input layer, hidden layer or layers, and output layer. Each node in the previous layer

is connected to each node in the next layer. Each of the connections involves an activation function. MLPs are fundamental in supervised learning tasks and can approximate any continuous function, making them versatile tools in machine learning[48].

Following the MLP, one-dimensional Convolutional Neural Networks (1D-CNNs) were employed. These networks leverage the local spatial correlation of input features, making them suitable for tasks with sequential or temporal data[70]. They use convolutional layers with filters similar to image processing, such as Sobel. However, each of the channels is the result of a different convolution. Also, convolutions are trainable filters, unlike image processing filters. With proper training, these operations help to detect patterns from simple to complex. When this transition to complex patterns happens, the help of the pooling layers is used. They are mostly used and effective for image classification or recognition tasks[70].

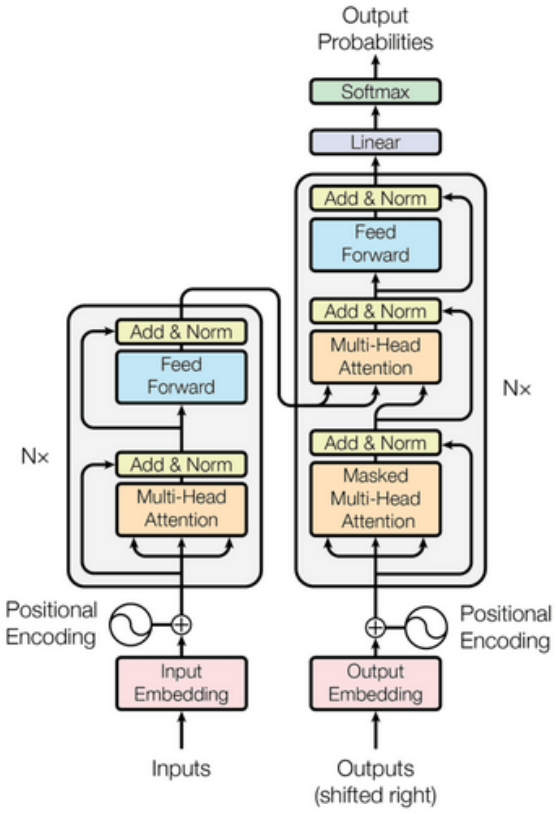


Figure 4.2: Sample Transformer architecture.

Residual Networks (ResNet) are a type of deep neural network architecture introduced to address the vanishing gradient problem commonly encountered when training very deep networks. ResNet uses residual blocks, which are designed to allow the model to learn residual functions with reference to the layer inputs instead of attempting to learn unreferenced functions (figure 4.1). These residual blocks contain shortcut connections that skip one or more layers, facilitating the direct flow of gradients through the network. This

architecture significantly mitigates the degradation problem, where adding more layers to a plain network leads to higher training error by ensuring that deeper models can be trained more effectively.

ResNet's innovative design enables the training of substantially deeper networks, leading to remarkable improvements in various tasks such as image classification, object detection, and semantic segmentation. The ability to train networks with hundreds or even thousands of layers has set new benchmarks in these domains, showcasing ResNet's robustness and flexibility in handling complex patterns and features in data[71].

In addition to its widespread use in two-dimensional applications, ResNet's architecture has been adapted to one-dimensional (1D-ResNet) scenarios, making it suitable for tasks involving sequential data such as time-series analysis. The incorporation of residual connections in 1D-ResNet enhances its performance by retaining both small and large patterns within the sequential data, thus capturing intricate temporal dependencies and improving predictive accuracy. This dual capability to handle both spatial and temporal data intricacies underscores ResNet's versatility and effectiveness in a broad range of deep learning applications[71].

Lastly, various Transformer setups were explored. They are a specific type of neural network developed for sequential data. It is mostly used for natural language processing tasks. Instead of processing the data sequentially from left to right or otherwise, they check which elements in the sequence have more importance simultaneously by using a self-attention mechanism. The base Transformer model was implemented, followed by versions incorporating locational encoding, convolutional encoding, and a hybrid of both. Transformers, renowned for their attention mechanisms, excel in capturing long-range dependencies and have shown remarkable success in various domains[72]. Incorporating locational and convolutional encodings aimed to augment the model's ability to understand positional information and local feature extraction respectively[73]. A sample Transformer architecture can be observed in Figure 4.5.

Each of these models was meticulously trained and evaluated, contributing to a comprehensive understanding of their strengths and limitations in the context of our specific application.

4.2 Alternative Dataset Trials During the Test of Deep-Learning Models

The final validity check of dataset generation was conducted using deep learning models, creating a comparative analysis between various approaches. Initially, the models were trained on samples with unscaled objects. As detailed in the previous section, the first dataset generated presented issues with distinctiveness, impacting the models' ability to learn effectively. This issue is visualized in Figure 4.3.

During the training phase with this initial dataset, the models exhibited varying behaviors, yet none surpassed random performance on the validation set. Both the Multilayer Perceptron (MLP) and Transformer models failed to memorize the data. To diagnose potential implementation issues, the gradients were inspected and found to be updating correctly. Despite this, performance improvements were not observed, suggesting the models struggled to learn the

complex patterns within the data[14].

Conversely, the convolutional and ResNet models managed to overfit the entire training set, but their validation performance remained at random levels. This indicated that while these models could capture the training data, they failed to generalize to unseen samples. Additionally, Long Short-Term Memory (LSTM) networks were evaluated, but their performance mirrored that of the other models, failing to memorize the data despite the presence of sequential information[74].

These results underscored a critical issue with the dataset: its lack of distinctiveness and the models' inability to grasp underlying patterns. Consequently, this necessitated the generation of a new dataset designed to enhance the distinctiveness and complexity of the samples. This new dataset aimed to provide clearer, more distinguishable patterns that deep learning models could effectively learn and generalize from.

Moving forward, the new dataset will be rigorously tested across the same range of models, including MLP, Transformer, Convolutional Neural Networks (CNNs), ResNet, and LSTM, to validate its effectiveness and improve model performance. This iterative process is crucial for refining the dataset and ensuring the robustness and reliability of the deep learning models employed in this research.

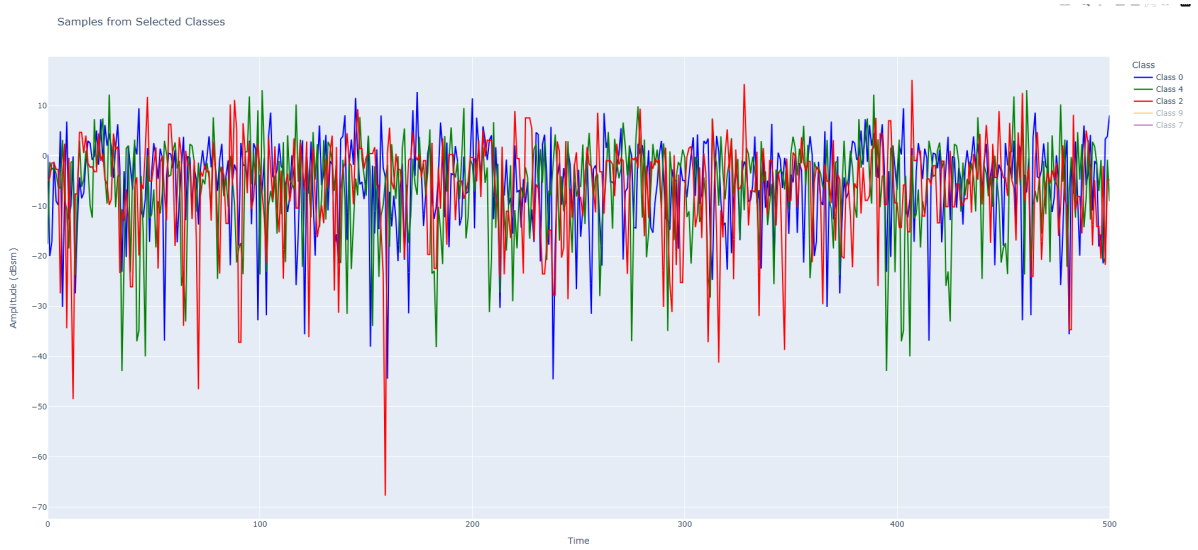


Figure 4.3: Sample plots of different classes with respect to dBsm before rescaling.

The data is structured in the format "classname/ObjectID," where each class contains a minimum of five objects for the initial trial phase. The "ObjectID" files are stored as .mat files, each with 512 columns and 2000 rows. In this configuration, each column represents a timestep, while each row corresponds to a sample reading associated with a specific object.

To ensure the samples were not over-representative, a simple mathematical equation was employed in equation 4.1. Out of a total of 522,728 possibilities, only 2000 were selected, resulting in a representative rate of 0.0038 (equation 4.2). This low rate alleviates concerns about over-representation in the dataset, ensuring a diverse and balanced sample selection.

$$AllPossiblesamples = Numberof Directions * Numberof possiblestartingpoints \tag{4.1}$$

$$522.728 = 8 * 65.361 \tag{4.2}$$

The .mat file format is not natively readable within the base Python environment. To address this, the external library SciPy was integrated into the project to facilitate the reading of .mat files. The `scipy.io.loadmat` method offers various options for converting the content of these files, with dictionaries and data frames being the most common formats. For this project, the variable containing radar samples was extracted as a dictionary pair. Subsequently, this data was assigned to another variable and converted into a PyTorch tensor, ensuring compatibility with the training loop[75].

This conversion process is crucial for efficient data handling and manipulation within the deep learning pipeline. By leveraging PyTorch tensors, the data can be seamlessly integrated into the model training and evaluation phases, facilitating streamlined workflow and enhancing computational efficiency.

The framework chosen for the implementation is PyTorch Lightning. PyTorch Lightning is an open-source library that serves as a high-level wrapper around conventional PyTorch, facilitating a more organized and streamlined coding experience. This framework offers several advantages, including the elimination of repetitive tasks related to data loading, logging, and checkpointing. Additionally, PyTorch Lightning is highly compatible with CUDA and other GPU accelerators, which enhances computational efficiency.

The framework's popularity can be attributed to its ability to accelerate development, maintain a cleaner code structure, provide an easy environment for experimentation, and offer scalability. By increasing reproducibility and flexibility, PyTorch Lightning significantly boosts research productivity, allowing for more efficient experimentation and implementation[76].

Three main classes play a crucial role in creating the model. The `CustomDataset` class accepts a root directory and loads every data file from the specified folder, assigning all subfolder names as class labels. It then processes the samples by converting their magnitudes to decibels. This conversion involves squaring all values in the dataset, taking the `log10` of these values, and then multiplying by 20. This decibel conversion serves two primary purposes: it makes the data more interpretable for humans and compresses the values into a narrower range, aiding the model in pattern recognition. The data and labels are then stored for further use within the class object.

4.2.1 Implementation of Deep Learning Environment

The `CustomDataset` class includes two methods adhering to the PyTorch Lightning convention. The `len` method returns the length of the dataset, equivalent to the number of data samples. The `getitem` method retrieves a data sample from the dataset object and ensures compatibility by converting the sample to a `torch.FloatTensor` for both data values and labels.

The `RadarDataModule` class manages the operations related to the training and validation sets, feeding them to the model as required by the PyTorch Lightning module. It comprises three methods to meet these needs. The `setup` method splits the initial dataset into training and validation sets. The `train_dataloader` and `val_dataloader` methods return the training and validation sets, respectively, as defined by the `setup` method.

The final class is the model itself, which varies according to the chosen architecture. Most architectures include a `forward` method that executes a full forward pass through the model. The `training_step` method handles the entire training loop, while the `validation_step` method assesses the model's performance on the validation set without updating the weights. The `configure_optimizers` method defines the optimizer, typically the Adam optimizer, as specified[77].

4.3 Obtained Results and Comparison

Previous results with the unscaled dataset primarily yielded random accuracy. After thorough analysis, a new scaled dataset was developed to address these issues. Instead of using all the previously tried architectures, only the promising ones that can memorize patterns in the previous dataset are chosen. To name them, the one-dimensional ResNet and Transformers, with various setups, were selected for further experimentation.

Initially, both the one-dimensional ResNet and Transformer models managed to memorize the unscaled dataset but failed to perform better than random. Multiple setups were tested with the new scaled dataset, particularly focusing on Transformers. The first setup involved a basic Transformer model with 8 layers and 4 attention heads. Given the input sequence length of 512 tokens, a Transformer with 8 layers and 4 attention heads is a balanced choice. The model is not excessively large, which helps in preventing overfitting and aids in generalization. Additionally, 4 attention heads allow the model to capture various patterns and correlations within the sequence, though the adequacy of this configuration depends on the specific task complexity and data characteristics. The rationale behind this configuration was that, given the time-series nature of the signal readings, Transformers could potentially capture relationships between previous and subsequent timesteps. This base Transformer achieved an accuracy of approximately 40%, indicating an improvement over random performance but still falling short of the desired goal.

Subsequently, a positional encoding approach was employed to provide additional information. The data dimensionality was increased from 1 to 8, with the new dimensions populated using sinusoidal and cosine functions. Despite this enhancement, there were no statistically significant improvements in accuracy performance.

Another encoding method explored was the use of convolutional layers, inspired by architectures like U-Nets, which are structured as encoder-decoder systems. In the final Transformer trials, a similar approach was adopted. Three layers of convolutions were stacked, incrementally increasing the dimensions to 256 over three steps. This transformed each data sample into a shape of 256x512. Feeding these new data samples into the same Transformer architecture resulted in a notable performance boost, with accuracy increasing to 60% (figure

4.5).

These results underscore the importance of dataset scaling and appropriate model architecture selection in enhancing performance. The iterative process of adjusting model configurations and dataset attributes is crucial for achieving optimal outcomes in deep learning tasks[72, 71].

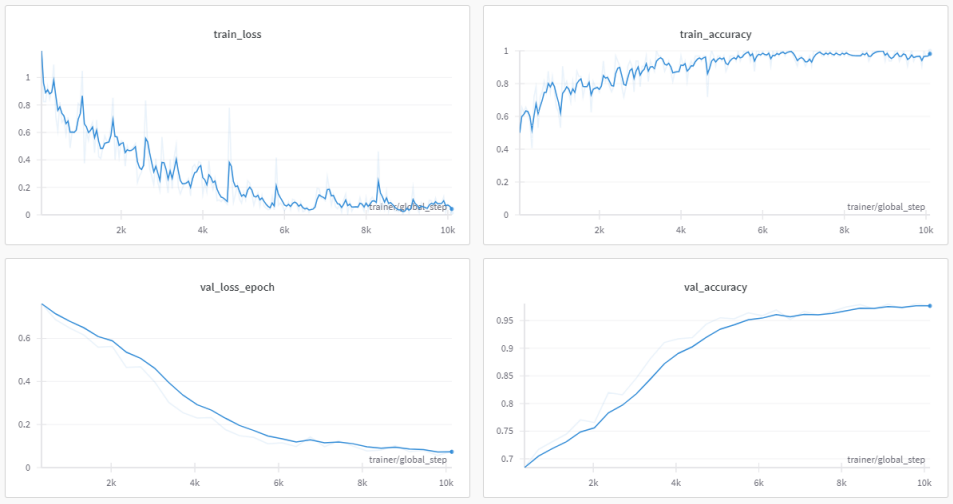


Figure 4.4: Loss drop and accuracy increase during the ResNet Training, shows the case of good generalization.

Another deep learning architecture that received significant focus is the Residual Network (ResNet). ResNet is characterized by its use of skip connections, a mechanism that feeds precalculated data into subsequent layers, effectively mitigating the vanishing gradient problem and allowing the network to learn identity functions. Although ResNet can support very deep networks, the version utilized in this study was ResNet8, a relatively shallow model. The choice of ResNet8 was driven by its effectiveness in learning identity mappings and its ability to handle vanishing gradients, rather than the depth enabled by skip connections.

In adapting ResNet for this study, the number of channels in the 1D convolutions was increased up to 512. While ResNet is typically designed for 2D image processing, modifications were made to tailor its architecture to 1D data. After extensive tuning, ResNet1D achieved an impressive accuracy of approximately 97% on the scaled dataset (figure 4.4). This high level of accuracy demonstrates the efficacy of both the simple data generation pipelines and the deep learning networks employed.

The success of ResNet1D in this context highlights the potential for applying deep learning models to more complex and realistic versions of the problem. This realization prompted the next phase of the thesis, which involves generating data in a manner that more closely reflects real-world conditions. This progression underscores the continuous evolution and refinement necessary in deep learning research to tackle increasingly sophisticated challenges[71, 78].

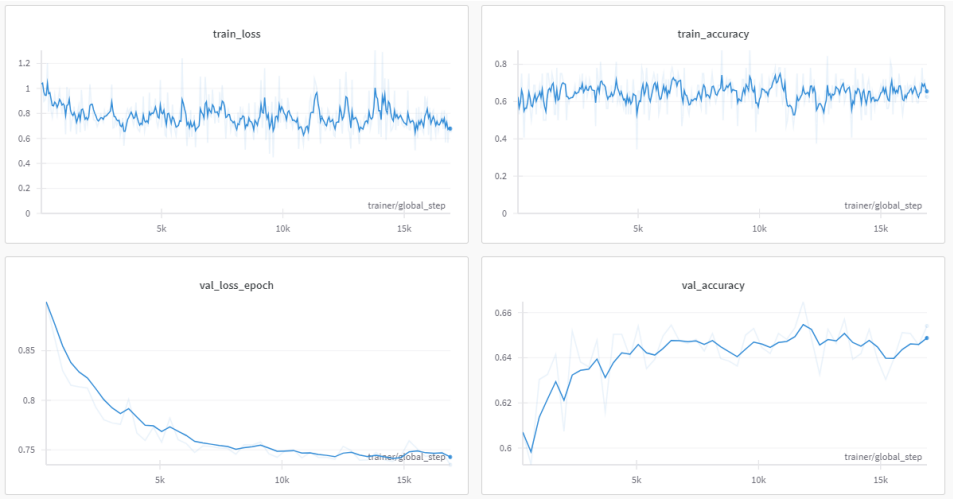


Figure 4.5: Validation loss drop and validation accuracy in Transformer, shows model learns but not ideally.

5 Complex Data Generation Simulation

The absence of a suitable dataset in the literature necessitated the creation of a novel solution for this project. Ideally, datasets are generated through extensive efforts and meticulous annotations, requiring a realistic setup to ensure accuracy. However, this process can be costly and complex. An alternative approach that balances cost reduction and complexity involves creating middle-ground solutions. This section aims to provide an explanation of such an approach. By excluding hardware and environmental constraints, we can focus on the essential aspects and simulate them as realistically as possible. This method allows for the exploration and testing of potential designs, problems, and solutions before developing a real prototype.

For the complex approach in this thesis, a simulation environment considering basic radar parameters realistically was created using Matlab. Additionally, a Python environment was developed to control the Matlab simulation. This setup allows for the generation of realistic radar returns rather than relying on hypothetical ones. The simulation environment considers multiple actors, providing a more comprehensive and realistic scenario.

The simulation begins with the definition of a pulse radar, configured with parameters that reflect a possible real-world setup. Following this, a three-dimensional object is defined with a volume specified by a bounding box and a Radar Cross Section (RCS) matrix that accounts for the object's material and shape. Although some aspects are not included in the current setup, the framework is designed to be scalable.

The radar system defined in this scenario is one-dimensional, featuring a single emitter and receiver. While it is possible to employ more complex radar structures that provide additional information such as azimuth and elevation angles along with range. This would require the expertise of radar specialists to configure the simulation environment accurately. Moreover, incorporating these additional parameters would alter the resulting "data cube," necessitating adaptations in the analysis tools used for processing the data.

By establishing a robust simulation environment, we can effectively model complex radar scenarios and generate realistic datasets. This approach not only mitigates the challenges associated with hardware and environmental constraints but also provides a versatile platform for testing and validating various designs and algorithms before physical implementation. This iterative process is crucial for refining the system and ensuring its efficacy in real-world applications[49].

5.1 Simulation and Radar Parameters

As mentioned earlier, the simulation was developed within a Matlab environment. However, the base Matlab environment alone was insufficient to build all the necessary simulations. To address this, six additional Matlab packages were integrated into the project. These packages included:

- Antenna Toolbox
- DSP System
- Radar Toolbox
- Signal Processing Toolbox
- Phased Array System Toolbox
- Communications Toolbox
- Parallel Computing Toolbox

The radar system designed for this simulation needed to detect and extract features from objects over long ranges. Inspired by Mari Ramirez-Torres's work on "Technological Developments for a Space-borne Orbital Debris Radar at 94 GHz," the radar frequency was set at 94 GHz. This frequency ensures that objects ranging from one to ten centimeters remain within the optical reflection region, maximizing the Radar Cross Section (RCS) values[15].

The simulation process begins by calling the `runSimulation` method from the Matlab engine via Python. Since Python handles object labeling and data saving, this method only requires the RCS matrix and the bounding box dimensions of the object. The setup script is then executed, defining essential environmental variables such as the speed of light and radar frequency, along with other interdependent radar variables. An iteration cycle logic was constructed using each pulse cycle as a unit. Unlike the previous pipeline, pulses and seconds are interchangeable in this setup, allowing the simulation to run in real-time.

In Matlab, many components are defined as objects, necessitating their prior creation. Therefore, the scene is initialized as a radar scenario. The radar is instantiated as an object of `radarTransceiver`, configured with its maximum range and field of view. Other radar-dependent variables defined earlier in the script are assigned to this object. The radar's waveform is defined as a `LinearFMWaveform` and linked to the radar object. Notably, there are two methods for obtaining range-Doppler images: using a Matched Filter or Fast Fourier Transformations. If the Matched Filter is preferred, the waveform coefficients must be saved at this stage.

Creating objects in Matlab does not imply their instantiation. They can remain abstract entities, establishing abstract connections with assigned or unassigned variables. Upon completion of the `setupScene` script, control returns to the `runSimulation` script, where the flying object and its associated parameters are instantiated.

At the end of the method, the simulation script that encapsulates the platform environment is executed before returning the variables. Initially, two boolean parameters are defined to control the plotting of IQ signals and range-Doppler images. Subsequently, the predefined environment variables are iterated for each timestep. To minimize confusion, all non-calculative parameters are explicitly stated beforehand. These parameters are then iterated, and the results are computed based on their current states. Each calculation is appended to a matrix that stores pulse reading values according to the sampling rate. Finally, a `phased.RangeDopplerResponse` object is created and executed with the data matrix to generate the range-Doppler matrix.

The resulting IQ and range-Doppler values are inherently complex. However, plotting complex values in 2D images is not intuitive, so a conversion to magnitude is performed before plotting. The saved variables are retained as complex values to prevent data loss. After executing each script, all variables are cleared except for the final result to conserve memory. This streamlined approach ensures efficient memory usage while maintaining the integrity of the simulation data.

5.2 Working Principle of Environment and Radar

For the pulse radar, there are variables or concepts that are highly essential to its working principle. Some of them are indeed the parameters that are mentioned as variables relevant to each other. A pulse radar cycle consists of a few components. To mention some of them, pulse repetition frequency (PRF) and pulse repetition interval (PRI) are two of those. They are mathematically inverse of each other. Pulse repetition interval represents a whole cycle as a time unit. Meaning from the start of the creation of the pulse and its return when the listening phase is over. This whole cycle can be observed in Figure 5.1. Pulse repetition interval is highly related to the effective range of the radar[79][49][48]. The listening time where the pulse is off directly defines the radar's maximum range. When the listening time is multiplied by the speed of light and divided by two, it will travel forth and back. It gives the possible maximum range of the radar regarding physical constraints. Most of the time, due to practical reasons, it is expected to be lower[53].

Pulse width defines how long the pulse emitting process takes as time units. There are some conventions for the ratio between pulse width and pulse repetition interval such as it could be 1/8 times or 1/16 times. But this can change according to needs. Pulse width is also directly correlated to how much information each pulse contains. The wider the pulse, the longer the collision between the pulse and the object, hence more information it contains. On the other hand, it also defines the minimum healthy measurement distance from the radar. Wider pulses reduce the overall observation area by making the minimum range larger. This comes from the possibility of a reflection while the pulse is still emitted. If there is an object that is 5 microseconds away with light speed, and our pulse width is 11 microseconds, the first reflection would reach the receiver at the tenth microsecond. However, since the pulse width is 11 microseconds, it is still in the emitting process. So the returned reflection is not visible or readable by the receiver. From this example, the minimum efficient range of a radar

regarding pulse width can be calculated as half of the pulse width times the speed of light in microseconds or relevant time units. If a radar operates with a 10-microsecond pulse width, then its minimum range is 5 times 300 meters (approximate speed of light in microseconds) 1500 meters[10].

Another boundary for the efficient reading distance is the upper limit, the maximum range. In the simulation, the maximum range is defined as an independent parameter. So even if the radar is physically capable of detecting an object, if it is further than the max range value, it will be ignored. Actually, any reading, even noise, will be ignored that is further than the maximum range. This is made to reduce computation complexity. Normally, as mentioned before, listening time defines the maximum efficient range. And the combination of pulse width plus listening time creates a full cycle also known as pulse repetition interval. For the simulation environment, each iteration is bound to a PRI. This means for each iteration in the for loop a full cycle happens, the response is read and recorded[53].

Antenna beamwidth is another property of the radar. It represents the angular spread of the detection area. It creates a cone-like area with the angle and range as the height. The regular convention is picking the angle that is at least at a point that has half of the peak power. As can be deduced from the cone metaphor, with a smaller angle comes a narrower cone and a smaller beamwidth. Two approaches, picking large or small, bring different advantages. A narrow beam width provides higher resolution and accuracy for the detected object's location. But will lose the coverage area for one scan. With a wider beamwidth, it would be possible to scan a larger area at the same time, but accuracy would drop. Depending on the need, both approaches are commonly used. If accuracy is the key in the task, like air traffic control or live missile tracking a narrower beamwidth is more beneficial. For the applications like weather prediction a broader scan is more beneficial hence wider beamwidth is used. There are also hybrid approaches, they are first using larger beamwidths to detect the initial objects then switch to narrower ones to track the object more precisely[53, 10].

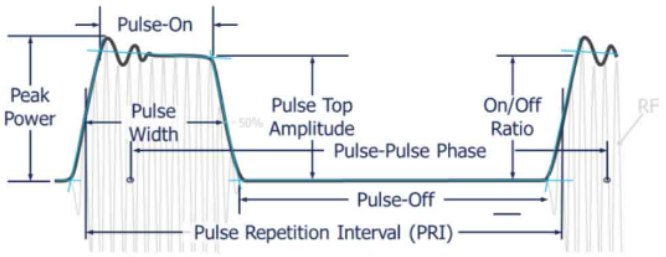


Figure 5.1: Example work cycle of a pulse radar.

In the universe, waves are continuous concepts. During the digitization of any continuous term, discretization is needed. The sampling rate determines how frequently the radar takes readings from incoming continuous signals. It is an essential concept for capturing the reflection echo accurately and serves as the endpoint of the physical components of the radar, thus remaining a critical part of the reading process. According to the bottleneck principle,

low performance at this stage results in poor readings overall. The information read to determine the characteristics of the object, such as shape in our case, is hidden within the signal variations. If the sampling rate is insufficiently high, all of that information will be lost, and Doppler shift and range information might also be compromised[80].

The Nyquist-Shannon theorem is fundamental for perfectly recovering signals from sampled data. The theorem states that the sampling frequency must be at least twice the highest frequency within the signal[81]. In pulse radars, the highest frequency component is related to the pulse width. Aliasing can occur if the sampling frequency is lower than the Nyquist rate, meaning the samples do not correspond exactly to one continuous function. As a result, multiple signals can be reconstructed from the same set of samples, leading to ambiguity. A similar scenario can be observed with helicopter blades in low-frame-rate cameras. While selecting a higher sampling rate than the Nyquist rate is not harmful in terms of reconstruction, it requires more memory to store the additional samples. If the primary task is only detection, a lower sampling rate may be acceptable. However, in most cases, it is preferable to choose a sampling rate above the Nyquist rate, even at the cost of increased memory and computation power[80]. Representation of a signal with various sample rates can be seen in Figure 5.2.

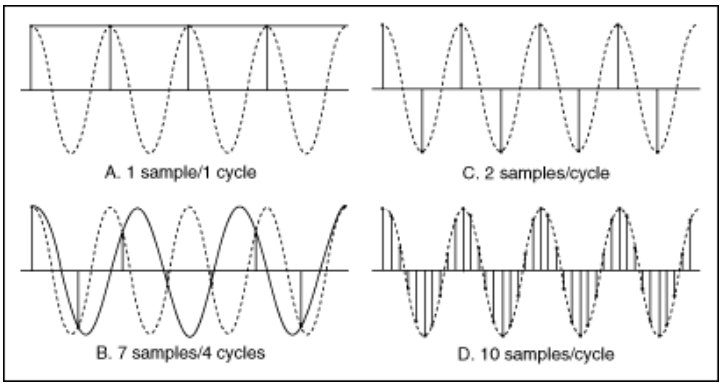


Figure 5.2: Same signal with different sample rates.

The final transmitted wave is a combination of slightly different complex waves. Pulse bandwidth defines how different those component waves are with respect to frequencies. Pulse bandwidth and pulse width are correlated with each other. The shorter the pulse width, the wider bandwidth needs to be used. The time-bandwidth product states that pulse width and pulse bandwidth are approximately constant for a defined pulse shape. If the pulse width is shorter, the frequency components need to be wider[53, 10]. Pulse bandwidth is important for range resolution, which evaluates how distinguishable two objects are at different ranges. Additionally, pulse bandwidth plays a critical role in pulse compression for improving the signal-to-noise ratio. A more complex radar setup is required when a wider bandwidth offers good range resolution, whereas a narrower bandwidth is more common for simpler applications[53].

5.3 Variables Defined in Simulation

Besides radar and its bound parameters, the simulation is more about the space scenario that is defined by the problem. The ultimate goal is to detect and classify debris with the returned signal. Therefore, the defined radar should be placed in an environment simulating a possible debris detection scenario. Creating a perfectly realistic simulation is an endless task, but certain unique features of the debris detection scenario distinguish it from other possible radar applications. This section discusses four of these characteristics and explains their relevance in the simulation.

Speed

Speed is the first feature implemented into the simulation. In space, there is no friction, which means objects maintain their speed constant most of the time. In low Earth orbit (LEO), orbiting target objects can reach speeds of up to 7.8 kilometers per second. In fact, a minimum speed is required for orbiting behavior, with each speed value corresponding to an orbit with a unique distance to the Earth's center of mass. However, the radar detection area in this scenario is not a hypothetical point. Various setups with different ranges are possible to create. Therefore, defining a constant speed would be unrealistic. It is possible to constrain the speed interval to specific boundaries. This approach is less generalized and requires calculating boundary speeds by considering possible detection regions of the radar and determining the maximum and minimum speeds objects can have in those orbits[82].

However, this approach, while complex to implement in a software environment, does not account for unusual cases such as explosions or objects entering or leaving the atmosphere. Additionally, from the radar's perspective, it is impossible to know the exact location of the radar and the object without using GPS triangulation, which, although feasible, is beyond the scope of this project. Therefore, if the radar is centered in the 3D simulation environment, the object's speed is only the relative speed. This simplification allows the speed to be zero, as an object and a radar could be in the same orbit moving at the same speed. In this case, the object would appear to have zero speed relative to the radar[83].

Rotation

Another important aspect of the simulation is the object's rotation. In space, objects tend to rotate, and they can rotate at any speed and angle as long as the object is not under the effect of another constant force. Since there are multiple ways of representing 3D axes or coordinates, there are also multiple ways to represent rotation. The azimuth-elevation approach is used to calculate Radar Cross Section (RCS), but for easier explainability and mathematical calculations, another convention is preferred here: quaternions. Quaternions are a method of representing 3D rotations and rotations around axes using four components(figure 5.3, which include a scalar and a three-dimensional vector, offering advantages in avoiding gimbal lock and providing smooth interpolation[84].

In this scenario, rotation is defined by two different approaches. Since the object's reflection

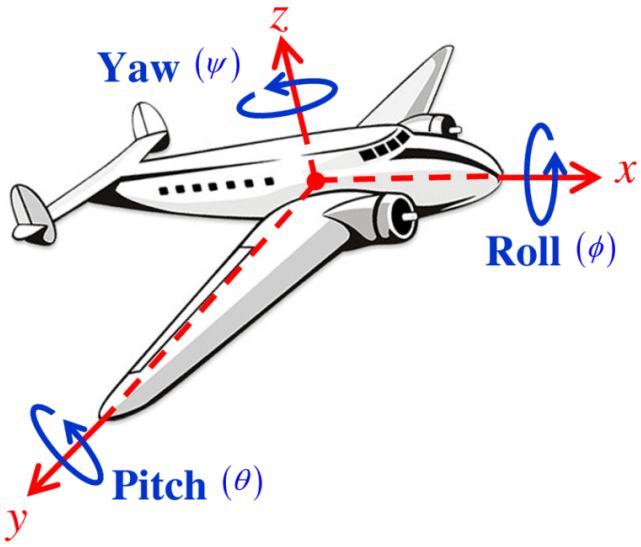


Figure 5.3: Quaternion Axes on a plane.

is derived from RCS and the set of RCS matrices is only 1-degree resolution, the first approach is to pick a random number from the set of {-1, 0, 1} and assign it to each axis, then multiply it with another random value in a way that avoids floating-point numbers. This provides a simple and computationally efficient method. In the more realistic approach, three random values between 0 and 1 are generated, their signs are assigned randomly, and they are scaled with another random value. This allows an object to rotate in any possible direction with any magnitude, increasing the realism of the simulation. However, since floating-point numbers are used, computation complexity can increase gradually[85].

Trajectories

The radar in this scenario has only one receiver, meaning it cannot directly predict the object's trajectory in the universe. However, the input data varies for objects moving closer or farther and moving towards the sides. Implementing trajectories is important to observe how the radar input behaves for signals entering or exiting the observation area.

To define the trajectories of objects, a circular area at the maximum range of the observation area is defined. This circle is described with a radius calculated as follows:

$$r = \tan\left(\frac{\text{antenna Beamwidth}}{2}\right) \times \text{maxRange} \tag{5.1}$$

With this radius, a circular area with radius r and centered at zero on the Y-axis, with r perpendicular to the Y-axis, is defined. Using this hypothetical circle, two points within it can be determined using the following formulas:

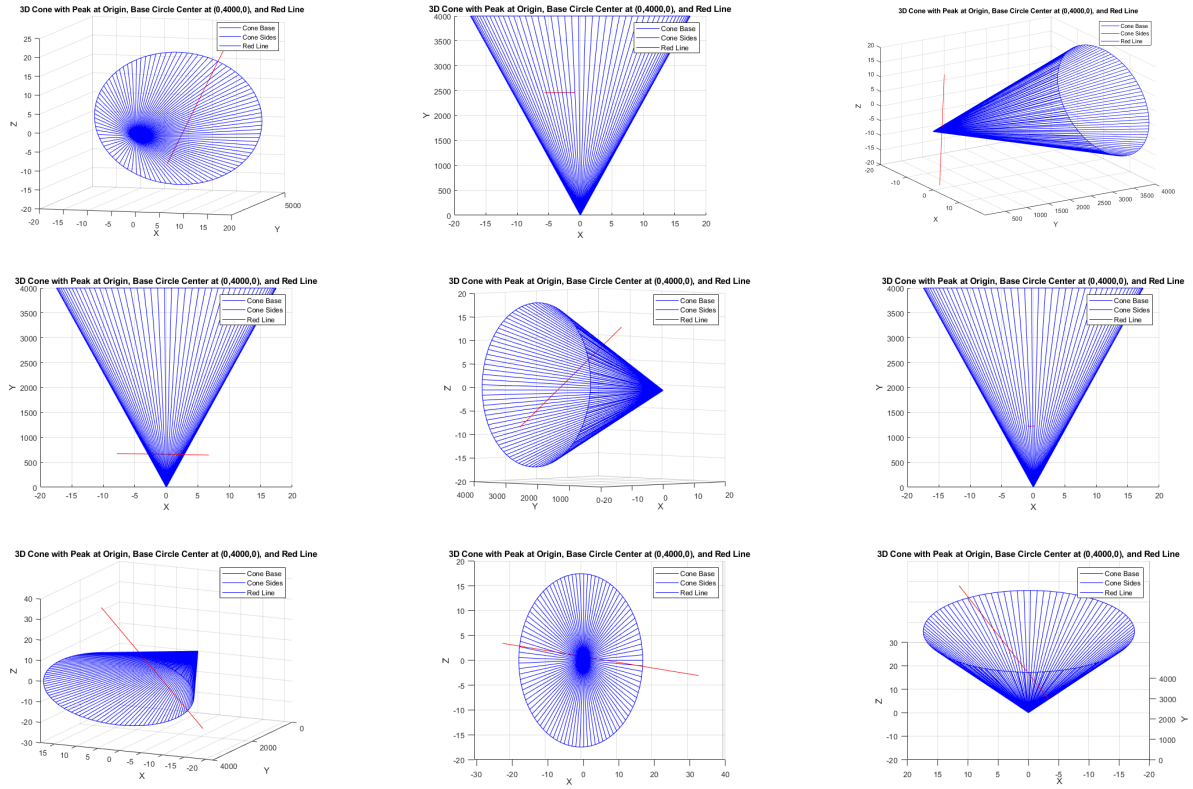


Figure 5.4: Sample trajectories of objects as red lines, blue cone represents radar scan area.

$$\text{SampleX} = 2r(\text{rand} - 0.5) \tag{5.2}$$

$$\text{randSampleZ} = \text{rand} \times \sqrt{\frac{r^2 - \text{SampleX}^2}{4r^2}} + 0.5 \tag{5.3}$$

$$\text{SampleZ} = 2r(\text{randSampleZ} - 0.5) \tag{5.4}$$

A check for the inequality $\text{SampleX}^2 + \text{SampleZ}^2 \leq r^2$ ensures that both points lie within the circle. This establishes a vector from the radar center to the hypothetical circle representing the furthest reading range. Not all objects have to be at the furthest point in the reading region, so this sampled vector is unit-wise multiplied by a value ranging from the minimum reading range that pulse width allows to 1. Assuming the reading range forms a cone, the bottom circle represents the hypothetical circle, and the height is the maximum range. The top of the cone is truncated due to minimum range restrictions, forming a remaining area used to obtain a sample point.

Converting a sample point to a trajectory involves transitioning from a point to a line that certainly crosses that sampled point. To achieve this, starting and ending points need to be defined. A random unit vector is created, and its subtraction from the sampled point provides the direction from the starting point to the endpoint. The sample point and direction are then added and multiplied by half of the pulses and the object's speed relative to the pulses. The endpoint is calculated by adding the starting point to the pulse repetition interval times the object's speed times the number of pulses. This method ensures the object crosses the reading area, though it doesn't necessarily start or end within it. Sample trajectories can be seen at Figure 5.4

The process described here aligns with standard radar trajectory simulation techniques, ensuring that the object's path through the radar's observation area is realistic and provides meaningful data for subsequent analysis[49].

Swerling

In Matlab, there is a function called `RCSSignature`; this method allows for defining features regarding the object, particularly its radar cross-section (RCS) characteristics. The primary purpose of this function is to assign the RCS matrix to the object, but it can also define the Swerling model with fluctuation model parameters. There are five different Swerling models that are commonly used, each covering different fluctuation scenarios.

Swerling 0 assumes perfect reflection, meaning no fluctuation occurs in the object's reflection. This model can be used to inspect other parameters in the simulation by bypassing fluctuation, but it detracts from realism. Swerling 1 varies RCS values with a Rayleigh distribution, similar to having only rough surfaces instead of smooth ones, thereby disturbing the reflection's clarity. Given the 3D object dataset's inherent lack of smooth surfaces, this Swerling feature is naturally covered.

Swerling 2 adopts a more realistic approach by using a chi-squared distribution on top of the RCS values. This behavior mimics reflections from multiple surfaces on the object, akin to

the reflections from a zig-zagged cardboard, where the return is a combination of multiple surface reflections. Swerling 2 is chosen for the simulation environment because it provides a more accurate representation of complex target reflections, which aligns with the nature of the debris detection scenario. Swerling 3 combines Swerling 1 with a constant component, and Swerling 4, similar to Swerling 2, emphasizes one main scatterer more than others. For complex targets, Swerling 2 is generally recommended due to its balance of complexity and realism.

It is important to note that none of the Swerling models might perfectly represent a realistic scenario. More sophisticated systems analyze the environment and estimate the appropriate Swerling model dynamically. This estimated Swerling model is then used to improve performance by more accurately simulating the object's reflective characteristics in varying conditions.

5.4 Assumptions and Constraints

During the simulation development, some goals were set. Creating a totally realistic simulation is not an achievable goal within the time period and limited resources. Therefore, a balance between detail and feasibility was necessary. After thorough discussions and planning, certain assumptions were made, and some limitations were left for future research and development. The first constraint is the developed environment. Matlab provides numerous frameworks and functions, but it is always possible to write all simulations from scratch, which could provide a much more detailed and better-represented simulation environment. Another constraint was the lack of a radar expert. Practical radars can be highly complex devices that far exceed the scope of a simple project. On top of these overall constraints, specific assumptions were made to finalize the project within a reasonable timeframe.

The first assumption concerns the radar's position and speed. As mentioned in the speed subsection, an assumption that allows ignoring absolute motion parameters was made. In ideal conditions, the simulation environment could be more dynamic with a freely moving radar, allowing for a more realistic definition of the object's speed. Additionally, the simulation does not consider objects moving in opposite directions in the relative speed calculation. Another assumption about the radar is that it always tracks the object, even at very high speeds. In realistic scenarios, the radar might lose track of an object if parameters are not developed accordingly .

The second set of assumptions relates to the object's size. Scaling in Trimesh scales the object into a bounding box with a maximum vertical orientation of 5 centimeters. However, this does not mean the object's size is exactly 5 centimeters. The longest line crossing the object could be longer than 5 centimeters in this scenario. Moreover, using a single scalar might not be informative enough for further predictions or uses. Volume and diagonal measurements could provide more detailed information. For instance, a cube with a 5-centimeter edge length and a stick with a 5-centimeter length are not equivalent in size. Furthermore, the simulation environment only recognizes the object as a bounding box. Even if reflections are accurate in practice, there is a possibility that a pulse might not hit the object due to its shape,

but it might hit the bounding box, causing a reflection. A good example is a cube with an edge length r and a globe with a radius r that fits within the cube. Not all rays hitting the cube necessarily hit the globe as well. The final assumption regarding sizes is that the radar size is neglected and represented as an abstract dot.

The frequency used is 94 gigahertz, as obtained from a radar base paper [15]. This setup is claimed to detect objects with a diameter of 1-10 centimeters and is designed for a 50-kilometer range. It is assumed to work within an 8-kilometer range as well, with many parameters derived from the cited paper. The main assumption about the frequency band is the difference between the base frequency and the frequency value at the end of the range. During travel, the pulse loses some of its frequency, so the end frequency value of the pulse at the end of the range needs to be calculated. All object's RCS needs to be recalculated with the obtained frequency. Using mid-frequency values instead of just the end frequency can increase simulation realism. Matlab interpolates between given RCS patterns if the distance falls into a space where RCS is not provided as an exact number. As expected, more values lead to better interpolation. However, to reduce computational complexity, it is assumed that frequency loss does not occur and that the pulse maintains the same frequency throughout its travel.

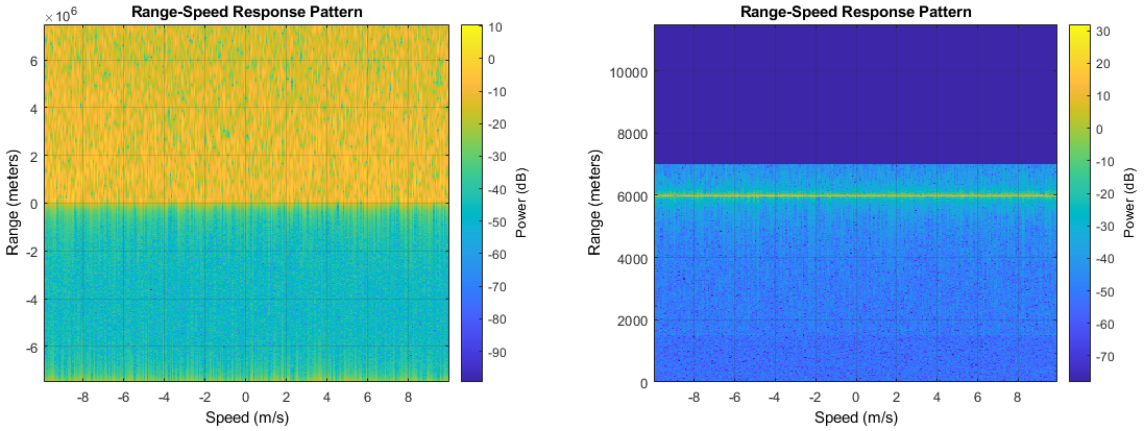
Another constraint arises from storage issues, further discussed in the data output section. Each receiver produces around 32 megabytes of data. Increasing the number of receivers would require much higher memory capacity. Since Matlab's wind turbine example demonstrated the feasibility of obtaining various signals with a single receiver, this setup was retained. Regarding the rotation of objects, a uniform distribution among possible angles is assumed. However, in reality, this is not the case. Due to the minimum energy arrangement, rotation at certain angles is more likely. This is influenced by two primary forces: gravity, which always pulls towards the center of mass, and the conservation of angular momentum.

Normally, space movements are ellipsoidal, but since the radius of the ellipsoid in this case is vast and our detection range is much smaller, all movement is defined linearly. Objects have predefined starting and ending points and move in a straight line. The radar scope is a cone, represented by an abstract cone slightly larger than the detection cone to ensure coverage. By knowing the speed and mass of the object, more realistic trajectories could be calculated. However, this would constitute a completely different project. Additionally, objects may move closer than the minimum range of the radar.

5.5 Filtered Matching and FFT Comparison

Fast Fourier Transform (FFT) and Match Filtering are two distinct methods for generating range-Doppler maps. Range-Doppler maps provide critical information regarding the distance of an object from the radar and its relative velocity. Both methods fundamentally differ in their approach and output, leading to various use cases and applications.

FFT is a widely used method for converting a signal from the time domain to the frequency domain[86]. When applied to the received signal from each pulse, FFT separates the returned signal based on its travel time. Travel time directly correlates with the object's range since



(a) FFT Doppler Image

(b) Matched Filter Doppler image

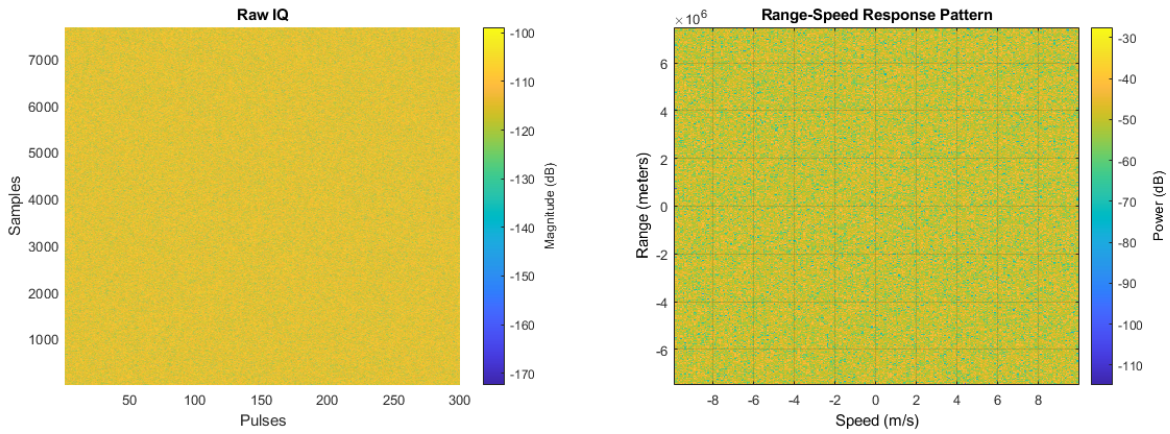
Figure 5.5: Comparison of same object with Matched Filter and FFT.

it takes longer for signals to return from more distant objects. FFT processes the received signals according to their range, estimating the target's location. To apply FFT, the raw signal is captured, and optionally, windowing and zero-padding are applied to the data. The mathematical application of FFT then shifts the time domain to the frequency domain, producing a complex spectrum. In this spectrum, higher frequency values indicate shorter travel times, while lower frequency values signify longer travel times[48, 87].

Match Filtering, on the other hand, requires knowledge of the generated signal's parameters. This method maximizes the signal-to-noise ratio by fitting the known signal shape to the received signal. The shape, in this case, is a replica of the transmitted waveform, also known as the matched filter[88]. Match Filtering is superior for determining both the range and velocity of the target compared to FFT. When applying Match Filtering, the transmitted waveform is used to design a matched filter. The received signal is divided into smaller pieces based on range resolution and radar characteristics. Correlation is then sought for all range bins with the matched filter. The obtained signal is fitted to a bin, providing an approximate range by the fitted bin[10, 89].

In summary, while FFT provides a straightforward method of converting time-domain signals to the frequency domain to estimate target location based on travel time, Match Filtering offers a more refined approach by maximizing the signal-to-noise ratio and fitting the known signal shape to the received signal for precise range and velocity determination. Each method has its unique advantages and applications depending on the specific requirements of the radar system. The differences between two method's output can be seen in Figure 5.5 with same object and parameters.

While FFT separates echoes based on their travel time alone, matched filtering also provides Doppler shift information, which can determine the relative speed of the object. However, FFT is faster in terms of calculation, and this speed difference becomes more pronounced as the waveform length increases. Due to its speed advantage, FFT is often used in real-time



(a) Empty IQ reading

(b) Empty RD reading

Figure 5.6: Empty readings for baseline.

processing applications. On the other hand, matched filtering is preferred in systems that require more accurate and detailed information. It is possible to combine both methods to leverage their respective advantages, particularly regarding range resolution. Initially, FFT was used to generate data in this project, but a switch to matched filtering was later made to enhance accuracy and detail in the simulation output.

5.6 Output: IQ Signal and Pulse Doppler Image

After the simulation is completed, it returns relevant parameters to the Python environment, specifically the IQ signal and range-Doppler map of the corresponding scenario. The IQ signal represents the raw signal data. Initial trials were conducted to observe the features of the simulation and the object. By not placing an object, the baseline properties of any returned IQ signal and range-Doppler map were understood. This provided insight into where the readings are irrelevant and devoid of information. It was observed that values around -100 decibels in the IQ signal represent empty readings. However, lower points than -100 dB were also present, indicating that the absence of a signal is not as flat as in ideal mathematical environments. The empty readings in both IQ signal and range-doppler image is given in Figure 5.6 for reference.

The two output variables are given as matrices. The dimensions of a matrix are defined by the number of pulses and the number of samples during the simulation. For the defined properties of the radar, the matrix size is (7671, 300). Here, 300 is the number of pulses, and 7671 is the number of samples taken during the work cycle. The number of pulses is determined by the user's preferences. This configuration is based on Matlab's wind turbine tutorial. The number of samples can be adjusted by changing any parameters relevant to the radar's work cycle, such as pulse width, sampling rate, or pulse repetition interval. Reducing these parameters decreases the matrix size in the number of rows.

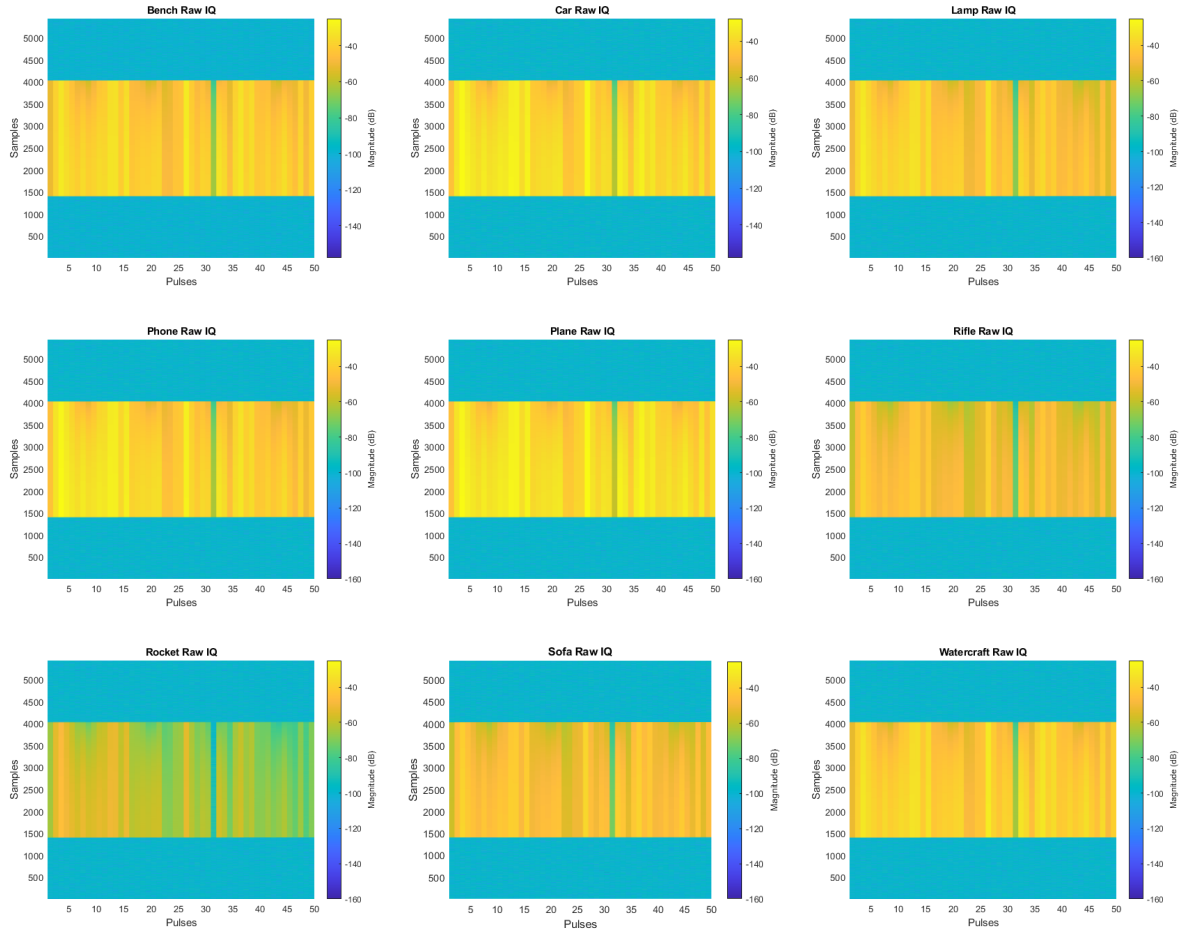


Figure 5.7: IQ signal comparison in between classes.

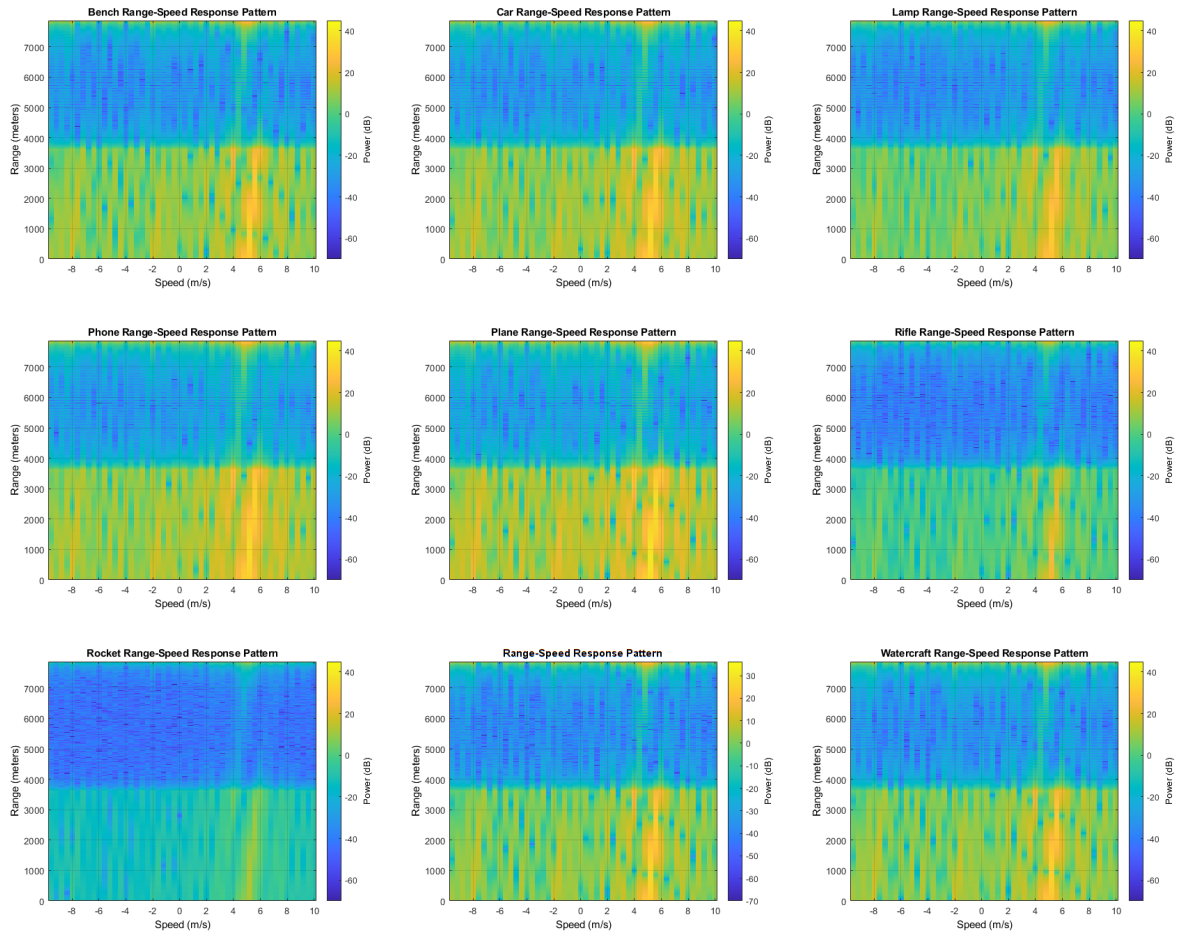


Figure 5.8: RD comparison in between various classes.

The matrix content consists of Matlab format complex numbers. Each cell contains both real and imaginary parts. It is noteworthy that plotting complex numbers can be challenging. Thus, for plotting, complex numbers are converted to real numbers by taking the squares of the real and imaginary parts, summing them, and taking their square roots. Each cell is converted to a scalar and plotted accordingly. However, the saved data retains its original complex form. Without processing, each data sample occupies approximately 30 megabytes, which is substantial. This issue will be addressed subsequently.

Before the Radar Cross Section (RCS) analysis, a series of trials were conducted to verify that different objects under the same conditions generate different signals. Sets of IQ and range-Doppler images were generated. Significant differences between classes were observed (figure 5.7 and 5.8). To confirm this, a series of subtractions (figure 5.10) among resulting output signals were performed. The potential uses of the data do not necessitate the complex data form, although this does not imply that the information is superfluous. For visual analysis or deep learning approaches, only the image format is required. For mathematical

analysis, the real part suffices as long as the value is preserved. Consequently, for convenience, the imaginary part is removed from the cells but added as a second dimension. Thus, the new data dimensions for the previous example become (7671, 300, 2). This conversion does not affect memory efficiency.

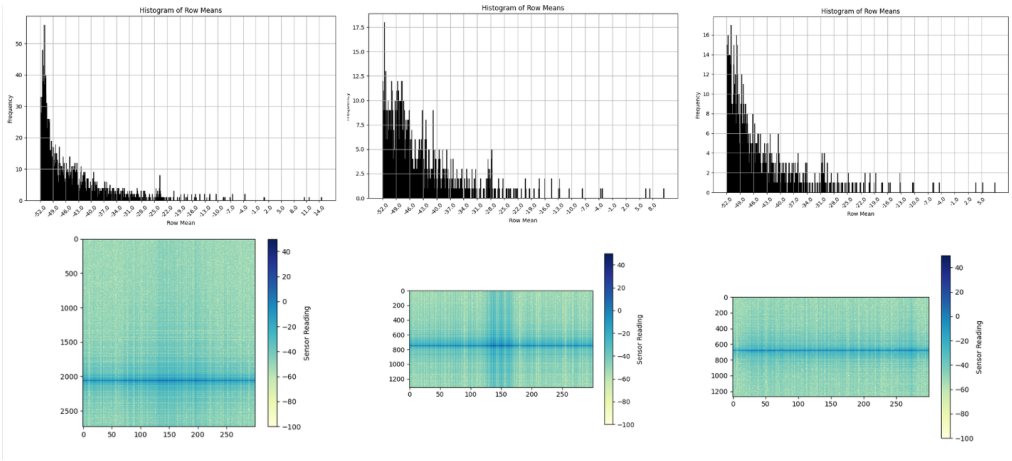


Figure 5.9: Histogram analysis of generated range-doppler images.

The challenge of observing the differences between the classes lies in the nature of the range-Doppler images. Firstly, these images are difficult to interpret, especially for non-radar experts. Secondly, the proportions of the matrix are not close to a square. The ratio between the number of rows to columns is 25, which remains substantial even when the plotting is rectangular. A possible solution to strengthen the hypothesis about the class differences is by utilizing histograms. From the filtered matched range-Doppler images, each row is obtained by its means, and a histogram is plotted to observe patterns (figure 5.9). For each plotted histogram, the majority of the rows had a value of around -50 with a Gaussian distribution. Given that most of the range-Doppler map consists of empty readings, -50 is assumed to be the noise value. The difference in patterns in histograms under the same simulation conditions was convincing enough to indicate that the signal returns for different classes are distinct.

An attempt was made to reduce the number of samples by changing the radar setup parameters. The tested parameters were a 9 megahertz pulse width and a 20 milliseconds pulse repetition interval. With this setup, each pulse had around 100 sample points left after removing the irrelevant empty readings. The number of pulses was also reduced to 100 for the same purpose. However, this idea was dropped when guidance from the supervisor suggested it might be unhealthy to make these changes due to insufficient expertise on radars. Another idea was to reduce the sampling rate, but it was not attempted since reducing below the Nyquist rate is highly discouraged.

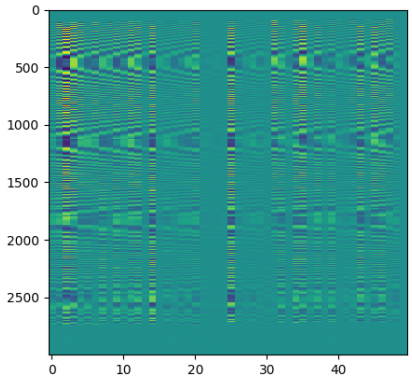


Figure 5.10: Substraction of a Car sample from a Plane sample to see if there are paternal differences.

5.7 Filtration of Output

The radar data being heavy causes many problems. The first of them is storage insufficiency. The memory amount authorized to use was 100 gigabytes; considering unprocessed data is 30 megabytes, the maximum storage can hold 3,334 samples at a time, which is not enough for a good sample set. It is important to mention that 30 megabytes is only for IQ signal; if range-Doppler maps need to be saved as well, the memory requirement would be doubled. Another problem that correlates to memory issues is calculation complexity. Considering the data is planned to be used in machine learning and various analyses, the size of the data will increase the complexity of such applications.

To solve this issue, a few strategies to reduce the memory requirement for each sample are derived. The data in range-Doppler maps is the most valued one for the case, as can be seen from the matched filter results; the values above the maximum range do not contain any information. The obtained complex signal is converted to amplitude to get rid of that part, and then the mean of the whole matrix is taken. The row's means are taken and appended to a list. When a row's average is lower than the matrix mean, that row is removed. This is the first filtering step applied to range-Doppler maps.

However, the data size was still not within the expected range, necessitating another filtering step. The next idea involved discarding the readings that have no object reading value. The outcome of histogram analysis is used for this purpose, taking -50 dB as the non-object reading signal value. By adding a margin, rows with average decibel values below -47 are removed. This reduced the number of rows, resulting in new shapes containing around 800-1200 rows. However, having different sizes for the dataset is not consistent. One sample might have a size of (862 300) while another might be (1032 300), which is problematic, especially for deep learning systems that require a static input size.

To address this inconsistency, a new filtering methodology was employed. Initially, the plan was to select rows with the highest means, for example, the highest 500 rows. However,

this approach could compromise data integrity. Instead, the method was adjusted to find the row with the highest mean and then select a constant number of rows above and below this peak. This ensured that only the signal-concentrated area was retained. After the second filtering, the final output had a shape of (600, 300).

6 Conclusion

This research thesis seeks a solution to the lack of readily available data or a method that can be used to generate data for space debris detection. It approaches the problem from multiple angles. Two different data generation techniques with different complexities and a deep learning solution for the pipeline data generated. Machine Learning is a tool to observe and mimic patterns in nature; by understanding the behavior set of a physical phenomenon, ML can suggest more suitable solutions rather than the existing solutions. Considering most of the current conventional signal processing methods are static and treat all the input equally, they may lack solutions for unique case scenarios or are sub-optimal in them. Also, extracting meaningful information from read values either requires high expertise or using stagnant methods that are hard to change for specific cases. Machine Learning can perform better and more dynamically with sufficient data and can reduce the need for expertise in domain. The ultimate goal was to utilize machine learning to bring an alternative to conventional signal processing methods in a faster and more accurate way. The main focus is given since data population is the first key aspect of any machine learning approach.

6.1 Findings

The watertight concept and conversion of ShapeNetCore objects became the first step of the solution. The combination of Trimesh and the Manifold repository led to the fixation of the objects with around a 96% success rate, which is not ideal, but considering any .obj file can be used, and ShapeNetCore has 51.300 unique objects, lack of population in any category won't occur. For the scenarios that require 100% of the objects, an already fixed set of .stl files can be used. During the experimentation phase, it was found that ShapeNet objects consist of multiple meshes for both different components of the object and the parts with different materials.

The developed RCS calculation method has proven to be faster than all other popular alternatives. The task requires the calculation of the illuminated region and reflection from 64.800 angles for 1-degree resolution. Written Matlab script achieved a performance of under 7 minutes for RCS calculation time in the "Linux Cluster". Under the same conditions, the mean RCS calculation time was 5.5 minutes, which is four times faster than another option, OptixRCS, with a performance of around 20 minutes. Since the result of the Manifold software causes non-smooth surfaces, it also introduces natural noise for extra realism. Additionally, utilizing the parallel pool with multiple cores or graphical processing units shows significant importance for the calculation time. Compared to the default single-core computation, it speeds up the calculation by four times, even with a 6-core basic local setup. The increase in

the calculation is a logarithmic function in the beginning, then with an increasing number of cores, it leans towards a linear function. The reason behind that is the Matlab-Engine starting time is considerably long, and job assignment to the cores with memory allocation could be the reason.

The analysis of the RCS results of the object was the most crucial part of the research. The lack of expertise in RCS initially led to a setback, but the analysis of parameters when calculating RCS provided a solution. Polarization, frequency, and output unit caused notable differences, but all the variations still held observable information regarding the nature of the object. However, the main parameter that provided the solution was the size or unit length of the object in the Matlab platform object. This means other variables are safe to change for different setups and experiments, but the size must always be within the desired range.

Another important point is the visualization of the resulting RCS matrix for objects. Since decibels are a logarithmic scale and the response between an empty signal and any surface is huge, it is highly important to constrain plotting values to some restriction standard. This was achieved by analyzing minimum and maximum values in all RCS values among objects. Using a new color scale made comparing objects and observing patterns among them easier.

With the simple data generation method, 54,000 samples were generated in under 5 minutes. The obtained data was checked with mean value analysis. There were no useful findings from this analysis. Even though the mean decibel values for the RCS matrix of the objects vary among classes, a slice within the matrix is not necessarily representative of the overall mean. This could be the case for objects that do not change signal reflection drastically on all surfaces. However, with complex objects, there could be some regions or peak points that are extreme cases and not representative of the overall mean value.

In addition, some deep learning models were applied to the obtained data. No model could identify any pattern in the data generated before the analysis. The hypothesis behind this is that even though Matlab doesn't provide a distance to the object in the platform but when calculating RCS when it's looked very closely at the object, reflections from no points of interest disrupt the data and introduce a super high noise that hides patterns. To validate this conclusion about models, gradient changes during training were analyzed, and no problem was found regarding the training cycle. Various deep-learning models were tried on the data generated after the analysis. Multi-layer perceptron, transformers, LSTMs, and ResNet are among the models that were implemented. After RCS analysis, hyperparameter tuning, and various alternative architectural approaches, Transformer and ResNet achieved the best results. During this phase, it was observed that the Transformer took more time to converge but with lower performance. Improving performance using sophisticated embedding approaches, such as an encoder, is possible. However, it still performed less well than ResNet with the embedding approach used in this project. On the other hand, ResNet was much more successful. It had a faster convergence rate and provided better generalization, which is crucial for accuracy. The overall goal of this phase of the thesis was to prove that deep learning architectures can classify differences between objects by only using raw signals. With the success of ResNet, a promising result was achieved, and the thesis leaned towards a more complex data generation approach.

Considering realism is one of the key concepts before deployment, especially for scenarios with very low error margins, space debris detection is one such scenario. All objects fly at very high speeds, making response time an important parameter. Detecting objects, recognizing, calculating trajectories, and initiating correction burns are processes that can take time, so in some of the scenarios, every second spent during the whole maneuver is highly valuable. Therefore, reducing detection and recognition time in more realistic scenarios without taking any risks in the simulation environment is a contributor to this domain. For this purpose, the simulation environment was developed using Python and Matlab. Detailed radar working principles and possible realistic scenarios were added to the environment. The health of the data and the coverage of the main components of the realistic scenario are ensured. Besides obtaining a complex simulation, alternative outputs are generated for different possible uses and projects. The simulation can generate both raw complex signal output and range-Doppler images. The memory issue is also pinpointed, and possible solution approaches are generated. From the simulation environment, observations were made to understand the nature of the nothingness and the object's response.

The simulation environment is capable of generating a sample in around 3 seconds if the relevant RCS matrix is ready. It can save both raw signals and range-Doppler images for the same simulation. Additionally, the radar setup can be changed and adapted to new scenarios.

6.2 Possible Future Research

In this section, we will discuss potential extensions to this thesis. Chronologically, mentioning the fixation of the objects to generate complex signal outputs, there are several points that could benefit from improvements. These extensions were not applied in this project due to time constraints or feasibility concerns regarding the outcomes.

First of all, a set of objects that are manifold by their nature can be generated. These objects can be tailored specifically for the project. Tools such as Blender or other 3D manipulation software can be utilized. Alternatively, 3D scans of the objects could be used to construct their software equivalents. Another improvement regarding object generation would involve introducing an alternative approach to the Manifold software. While the rough surfaces provided by Manifold are useful, they are not ideal for all scenarios, particularly when observing ideal conditions. An alternative fixation approach could be implemented on top of ShapeNet to achieve ideal conversion from .obj files to .stl files.

Regarding the RCS calculation, if the objects are fixed, the RCS of the objects can be measured using real RCS reading devices at different frequencies. These values can then be converted to the working frequency of the desired application. This approach would capture all the natural characteristics of the environment, resulting in the most realistic data. Additionally, material properties can be incorporated into the RCS calculation, considering that real-life objects are typically composed of multiple materials.

For the applied deep learning models, various enhancements can be made. More models and specific architectures can be employed to evaluate performance. However, a significant improvement would involve incorporating sophisticated deep learning models tailored

for different applications, such as autonomous driving, to adapt to this task. This might necessitate updates to the model input or the data generation output format. Furthermore, the robustness of deep learning models should be rigorously assessed. Tests involving random dropouts, sensor saturation, and data clutters can be conducted. Additionally, evaluating model performance under conditions of bit loss due to radiation in space can provide valuable insights. Finally, while the current models utilize raw signal output, further analysis and development using range-Doppler images can be explored.

The realism of the defined simulation can be further enhanced. By considering the radar's range and relevant parameters, the possible speeds of the objects can be restricted to a specific interval. Incorporating radar rotation to determine the azimuth location of the target object can further enhance realism, as this working principle is widely used in sonars. Defining trajectories as ellipsoids will also contribute to a more realistic approach. Additionally, regarding IQ signal and range-Doppler outputs, an in-depth analysis is needed to identify where the important data is represented, and more sophisticated filtering of the outputs can be performed.

Lastly, and more comprehensively, it is possible to generate collision simulations with objects and relevant materials defined beforehand. The result of the collision can be the objects that are spread around with a size smaller than 10 centimeters. With this kind of simulation, the objects would be the closest to the authentic. Then, those objects can be taken, and the RCS of those can be calculated. This would replace the ShapeNet and its manifoldization process.

These proposed future research directions hold promise for improving the accuracy, efficiency, and realism of radar signal processing and deep learning model applications, thereby advancing the field significantly.

List of Figures

1.1 Difference between mono-static and multi-static radars. 3
1.2 Steps of data generation pipeline. 4
1.3 Different types of radars and which information can be obtained from them with conventional methods. 5
1.4 Example work cycle of a pulsed radar: First a pulse is generated according to pulse width. Then pulse hits the obstacle and returns back. After the pulse generation, the antenna switches to the listening/receiving time to read received signals. 6

3.1 6 Sample objects from Shapenet. 14
3.2 Number of calculated Azimuth slices per minute with an increasing number of cores. 20
3.3 Calculation time in minutes per object with increasing number of cores. 20
3.4 Cars RCS slice with 0 elevation. 26
3.5 Different car objects with 0 elevation and 0-360 azimuth, unit as inches. 27
3.6 Planes RCS slice with 0 elevation. 27
3.7 Resolution comparison for Plane. 27
3.8 Average RCS Matrixes. 28
3.9 Sample round RCS matrixes. 28
3.10 Round RCS plot with new units. 29

4.1 Logic behind ResNet architecture. 32
4.2 Sample Transformer architecture. 33
4.3 Sample plots of different classes with respect to dBsm before rescaling. 35
4.4 Loss drop and accuracy increase during the ResNet Training, shows the case of good generalization. 38
4.5 Validation loss drop and validation accuracy in Transformer, shows model learns but not ideally. 39

5.1 Example work cycle of a pulse radar. 43
5.2 Same signal with different sample rates. 44
5.3 Quaternion Axes on a plane. 46
5.4 Sample trajectories of objects as red lines, blue cone represents radar scan area. 47
5.5 Comparison of same object with Matched Filter and FFT. 51
5.6 Empty readings for baseline. 52
5.7 IQ signal comparison in between classes. 53
5.8 RD comparison in between various classes. 54

5.9 Histogram analysis of generated range-doppler images. 55
5.10 Substraction of a Car sample from a Plane sample to see if there are paternal differences. 56

List of Tables

1.1 Radar Parameters 7

3.1 Amortization analysis 24

Bibliography

[1] *Union of Concerned Scientists*. 2024. URL: https://www.ucsusa.org/.

[2] *European Space Agency - Space Debris*. 2024. URL: https://www.esa.int/Space_Safety/Space_Debris.

[3] *NASA Orbital Debris Program Office*. 2024. URL: https://www.orbitaldebris.jsc.nasa.gov/.

[4] *Inter-Agency Space Debris Coordination Committee*. 2024. URL: https://www.iadc-home.org/documents_public.

[5] C. Li and C. Kang. "Development of nanosatellite constellations for space debris detection". In: *Acta Astronautica* 88.1-2 (2013), pp. 122–130. DOI: https://doi.org/10.1016/j.actaastro.2013.02.007.

[6] J.-C. Liou. "An assessment of the collision risk between operational spacecraft and space debris". In: *Advances in Space Research* 57.2 (2016), pp. 300–311.

[7] D. J. Kessler. "Collisional evolution of debris clouds". In: *Advances in Space Research* 11.12 (1991), pp. 63–70.

[8] T. Schildknecht. "Modelling and simulation of space debris impacts on spacecraft". In: *Acta Astronautica* 137 (2017), pp. 142–152.

[9] J.-C. Liou. "Orbits and collision probability of near-Earth objects". In: *Advances in Space Research* 38.9 (2006), pp. 2031–2036.

[10] M. I. Skolnik. *Introduction to radar systems*. 3rd. McGraw-Hill Education, 2008.

[11] A. e. a. Sanz-Garcia. "A constellation of cubesats for space debris detection and tracking". In: *Acta Astronautica* 148 (2018), pp. 25–33.

[12] A. Geiger, P. Lenz, and J. Uderstadt. "Are we ready for autonomous driving? The KITTI vision benchmark suite". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2013, pp. 3354–3361.

[13] W. Maddern, A. Geiger, M. Krishnan, D. Gurdan, and R. Fergus. "Semantic labeling in street scenes using conditional random fields and convolutional neural networks". In: *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*. 2017, pp. 3908–3914.

[14] Y. LeCun, Y. Bengio, and G. Hinton. "Deep learning". In: *Nature* 521.7553 (2015), pp. 436–444.

[15] M. Ramírez-Torres, M. Ferreras, C. Hernández, C. García-de-la-Cueva, M. Barba, G. Perez-Palomino, J. A. Encinar, M. Sierra-Castañer, J. Grajal, J.-L. Vázquez-Roy, and E. Rajo-Iglesias. "Technological developments for a space-borne orbital debris radar at 94 GHz". In: *2018 IEEE Radar Conference (RadarConf18)*. 2018, pp. 0564–0569. DOI: 10.1109/RADAR.2018.8378621.

[16] M. Landgraf, R. Jehn, and W. Flury. "Comparison of EISCAT radar data on space debris with model predictions by the master model of ESA". In: *Advances in Space Research* 34.5 (Jan. 2004), pp. 872–877. ISSN: 0273-1177. DOI: 10.1016/j.asr.2003.01.007. URL: http://dx.doi.org/10.1016/j.asr.2003.01.007.

[17] A. Murtaza, S. J. H. Pirzada, T. Xu, and L. Jianwei. "Orbital Debris Threat for Space Sustainability and Way Forward (Review Article)". In: *IEEE Access* 8 (2020), pp. 61000–61019. DOI: 10.1109/ACCESS.2020.2979505.

[18] B. Weeden. "2009 Iridium—Cosmos collision fact sheet". In: *Advances in Neural Information Processing Systems*. 2010, pp. 5–7.

[19] H. Miyamoto, M. Battisti, A. Belov, M. E. Bertaina, F. Bisconti, R. Bonino, S. Blin-Bondil, F. Cafagna, G. Cambiè, F. Capel, M. Casolino, A. Cellino, I. Churilo, G. Cotto, A. Djakonow, T. Ebisuzaki, F. Fausti, F. Fenu, C. Fornaro, A. Franceschi, C. Fuglesang, D. Gardiol, P. Gorodetzky, F. Kajino, P. Klimov, L. Marcelli, W. Marszał, M. Mignone, A. Murashov, T. Napolitano, G. Osteria, M. Panasyuk, E. Parizot, A. Poroshin, P. Picozza, L. W. Piotrowski, Z. Plebaniak, G. Prévôt, M. Przybylak, E. Reali, M. Ricci, N. Sakaki, K. Shinozaki, G. Suino, J. Szabelski, Y. Takizawa, M. Traïche, and S. Turriziani. *Space Debris detection and tracking with the techniques of cosmic ray physics*. 2019. arXiv: 1909.05601 [astro-ph.IM].

[20] M. Khalil, E. Fantino, and P. Liatsis. "Evaluation of Oversampling Strategies in Machine Learning for Space Debris Detection". In: *2019 IEEE International Conference on Imaging Systems and Techniques (IST)*. 2019, pp. 1–6. DOI: 10.1109/IST48021.2019.9010217.

[21] G. Muntoni, G. Montisci, T. Pisanu, P. Andronico, and G. Valente. "Crowded Space: A Review on Radar Measurements for Space Debris Monitoring and Tracking". In: *Applied Sciences* 11.4 (2021). ISSN: 2076-3417. DOI: 10.3390/app11041364. URL: https://www.mdpi.com/2076-3417/11/4/1364.

[22] J. Pandeirada, M. Bergano, J. Neves, P. Marques, D. Barbosa, B. Coelho, and V. Ribeiro. "Development of the First Portuguese Radar Tracking Sensor for Space Debris". In: *Signals* 2.1 (Mar. 2021), pp. 122–137. ISSN: 2624-6120. DOI: 10.3390/signals2010011. URL: http://dx.doi.org/10.3390/signals2010011.

[23] M. Y. Martin, S. L. Winberg, M. Y. A. Gaffar, and D. MacLeod. "The Design and Development of a GPU-accelerated Radar Simulator for Space Debris Monitoring". In: *Proceedings of the 2021 5th High Performance Computing and Cluster Technologies Conference* (2021). URL: https://api.semanticscholar.org/CorpusID:245426738.

[24] O. Daoust, H. Nayir, I. Azam, A. Lesage-Landry, and G. K. Kurt. *Tensor-based Space Debris Detection for Satellite Mega-constellations*. 2023. arXiv: 2311.11838 [eess.SP].

[25] M. Ulrich, C. Gläser, and F. Timm. *DeepReflecs: Deep Learning for Automotive Object Classification with Radar Reflections*. 2020.

[26] J. Rebut, A. Ouaknine, W. Malik, and P. Pérez. *Raw High-Definition Radar for Multi-Task Learning*. 2022.

[27] J. Giroux, M. Bouchard, and R. Laganiere. *T-FFTRadNet: Object Detection with Swin Vision Transformers from Raw ADC Radar Signals*. 2023.

[28] Y. Dalbah, J. Lahoud, and H. Cholakkal. *RadarFormer: Lightweight and Accurate Real-Time Radar Object Detection Model*. 2023.

[29] B. Yang, I. Khatri, M. Happold, and C. Chen. *ADCNet: Learning from Raw Radar Data via Distillation*. 2023.

[30] B. Major and et al. "Vehicle classification using raw radar data and ensemble learning". In: *IEEE Transactions on Intelligent Transportation Systems* (2019).

[31] O. Schumann and et al. "Semantic Segmentation on Radar Point Clouds". In: *IEEE Intelligent Vehicles Symposium (IV)* (2018).

[32] Y. Kim and et al. "Human Activity Classification Using Radar and Deep Learning". In: *IEEE Access* (2015).

[33] C. Decourt, R. VanRullen, D. Salle, and T. Oberlin. "A Recurrent CNN for Online Object Detection on Raw Radar Frames". In: *IEEE Transactions on Intelligent Transportation Systems* (2024), pp. 1–10. ISSN: 1558-0016. DOI: 10.1109/tits.2024.3404076. URL: http://dx.doi.org/10.1109/TITS.2024.3404076.

[34] H. Zou, Z. Xie, J. Ou, and Y. Gao. "TransRSS: Transformer-based Radar Semantic Segmentation". In: *2023 IEEE International Conference on Robotics and Automation (ICRA)*. 2023, pp. 6965–6972. DOI: 10.1109/ICRA48891.2023.10161200.

[35] A. Ouaknine, A. Newson, J. Rebut, F. Tupin, and P. Pérez. *CARRADA Dataset: Camera and Automotive Radar with Range-Angle-Doppler Annotations*. 2021. eprint: 2005.01456.

[36] Y. Dalbah, J. Lahoud, and H. Cholakkal. *TransRadar: Adaptive-Directional Transformer for Real-Time Multi-View Radar Semantic Segmentation*. 2023. eprint: 2310.02260.

[37] S. Kim and et al. "Real-Time Object Detection with 3D Convolutional Neural Networks for Range-Doppler-Azimuth Radar Data". In: *IEEE Transactions on Intelligent Vehicles* (2017).

[38] Y. Li and et al. "Multi-Sensor Fusion for Object Detection Using Radar and LiDAR". In: *IEEE Transactions on Intelligent Transportation Systems* (2020).

[39] B. Major and et al. "Long-Term Sequence Prediction with Recurrent Neural Networks for Radar Data". In: *IEEE Transactions on Neural Networks and Learning Systems* (2019).

[40] A. X. Chang, T. Funkhouser, L. Guibas, P. Hanrahan, Q. Huang, Z. Li, S. Savarese, M. Savva, S. Song, H. Su, et al. "ShapeNet: An Information-Rich 3D Model Repository". In: *arXiv preprint arXiv:1512.03012* (2015). URL: https://www.shapenet.org/.

[41] A. Xyz and A. Author2. "Subcategory in ShapeNetCore: A Large-Scale Dataset for Fine-Grained 3D Object Recognition". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2019.

[42] A. J. Christensen and K. Daniilidis. "Dense Pose Estimation in Real-Time Using SHREC-Net". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020.

[43] M. Dawson-Haggerty. *Trimesh*. Python library for loading and using triangular meshes. 2023. URL: https://trimsh.org/.

[44] J. Huang, H. Su, and L. Guibas. "Robust Watertight Manifold Surface Generation Method for ShapeNet Models". In: (2018). arXiv: 1802.01698 [cs.CG].

[45] *Prusa Slicer Documentation*. You can search for the latest Prusa Slicer user guide or online documentation.

[46] Autodesk. *Meshmixer Help*. URL: https://help.autodesk.com/view/MSHMXR/2019/ENU/.

[47] B. Foundation. *3D Print Toolbox Add-on Documentation*. URL: https://docs.blender.org/manual/en/latest/addons/mesh/3d_print_toolbox.html.

[48] M. A. Richards. *Fundamentals of Radar Signal Processing*. McGraw-Hill Education, 2014.

[49] M. I. Skolnik. *Radar Handbook*. McGraw-Hill Education, 2008.

[50] E. F. Knott, J. F. Shaeffer, and M. T. Tuley. *Radar Cross Section*. SciTech Publishing, 2004.

[51] N. C. Currie. *Radar Reflectivity Measurement: Techniques and Applications*. Artech House, 1989.

[52] D. K. Barton. *Radar System Analysis and Modeling*. Artech House, 2005.

[53] M. A. Richards, J. A. Scheer, and W. A. Holm. *Principles of Modern Radar: Basic Principles*. SciTech Publishing, 2010.

[54] R. F. Harrington. *Field Computation by Moment Methods*. 2nd. Chapter 2: The Method of Moments. IEEE Press, 1993. ISBN: 978-0863801392. URL: https://ieeexplore.ieee.org/book/3170.

[55] H. Harbrecht and J. Tausch. "Efficient algorithms for computing the solution of the second kind integral equation for acoustic and electromagnetic scattering problems". In: *Journal of Computational Physics* 230.21 (2011), pp. 8142–8156.

[56] A. F. Peterson, S. L. Ray, and R. Mittra. *Computational Methods for Electromagnetics*. Wiley-IEEE Press, 1998.

[57] L. Greengard and V. Rokhlin. "The fast multipole method for the wave equation: A pedestrian prescription". In: *Wave Motion* 9.4 (1987), pp. 325–375. DOI: 10.1016/0165-2125(87)90047-3. URL: https://doi.org/10.1016/0165-2125(87)90047-3.

[58] C. A. Brebbia. *The boundary element method for engineers*. Pentech Press Ltd., 1984.

[59] R. F. Harrington. *Time-Harmonic Electromagnetic Fields*. IEEE Press, 2001.

[60] W. C. Chew, J.-M. Jin, E. Michielssen, and J. Song. "Fast and Efficient Algorithms in Computational Electromagnetics". In: *Artech House antennas and propagation library* (1997).

[61] V. Rokhlin. "Rapid solution of integral equations of classical potential theory". In: *Journal of Computational Physics* 60.2 (1985), pp. 187–207.

[62] The MathWorks, Inc. *MATLAB and Simulink*. 2024. URL: https://www.mathworks.com/products/matlab.html.

[63] L. S. C. (LRZ). *Job Processing on the Linux-Cluster*. Documentation for job processing on the LRZ Linux cluster, covering cluster specifications, job submission, and management using SLURM. 2024. URL: https://doku.lrz.de/job-processing-on-the-linux-cluster-10745970.html.

[64] firepond. *OptixRCS: RCS Calculation Using the SBR Method and NVIDIA OptiX*. RCS calculation based on the Shooting and Bouncing Rays (SBR) method, leveraging NVIDIA OptiX. 2024. URL: https://github.com/firepond/OptixRCS.

[65] RedBlight. *RaytrAMP: RCS Calculation Using the SBR Method*. Shooting and bouncing rays method for radar cross-section calculations, accelerated with BVH algorithm running on GPU (C++ AMP). 2024. URL: https://github.com/RedBlight/RaytrAMP.

[66] T. Karras. "Hierarchical traversal for BVH ray tracing". In: *Journal of Computer Graphics Techniques (JCGT)* 1.1 (2012), pp. 1–10.

[67] A. Gerndt. *Pofacets4.5: POFacets - Physical Optics and Shooting Bouncing Ray Methods*. MATLAB Central File Exchange. Retrieved June 13, 2024. 2024. URL: https://de.mathworks.com/matlabcentral/fileexchange/50602-pofacets4-5.

[68] Altair Engineering, Inc. *Altair Feko*. https://help.altair.com/feko/index.htm. Accessed: 2024-06-13. 2024.

[69] G. E. Hinton, D. E. Rumelhart, and R. J. Williams. "Learning representations by back-propagating errors". In: *Nature* 323.6088 (1986), pp. 533–536.

[70] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. "Gradient-based learning applied to document recognition". In: *Proceedings of the IEEE* 86.11 (1998), pp. 2278–2324.

[71] K. He, X. Zhang, S. Ren, and J. Sun. "Deep residual learning for image recognition". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 770–778.

[72] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. "Attention is all you need". In: *Advances in neural information processing systems*. 2017, pp. 5998–6008.

[73] N. Parmar, A. Vaswani, J. Uszkoreit, Ł. Kaiser, N. Shazeer, A. Ku, and D. Tran. "Image transformer". In: *International Conference on Machine Learning*. PMLR. 2018, pp. 4055–4064.

[74] S. Hochreiter and J. Schmidhuber. "Long short-term memory". In: *Neural computation* 9.8 (1997), pp. 1735–1780.

[75] E. Jones, T. Oliphant, P. Peterson, et al. "SciPy: Open source scientific tools for Python". In: *https://www.scipy.org/* (2001).

[76] W. Falcon and T. P. L. team. *PyTorch Lightning*. `https://www.pytorchlightning.ai`. 2019.

[77] D. P. Kingma and J. Ba. "Adam: A method for stochastic optimization". In: *arXiv preprint arXiv:1412.6980* (2014).

[78] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. "Going deeper with convolutions". In: *Proceedings of the IEEE conference on computer vision and pattern recognition* (2015), pp. 1–9.

[79] B. R. Mahafza. "Radar Basics". In: *Principles of Radar* (2021).

[80] A. V. Oppenheim and R. W. Schafer. *Signals and Systems*. Prentice Hall, 1996.

[81] C. E. Shannon. "Communication in the presence of noise". In: *Proceedings of the IRE* 37.1 (1949), pp. 10–21.

[82] D. A. Vallado. *Fundamentals of Astrodynamics and Applications*. Springer Science & Business Media, 2001.

[83] M. C. Bryson et al. *Orbital Debris: A Technical Assessment*. National Academies Press, 2002.

[84] J. B. Kuipers. *Quaternions and Rotation Sequences: A Primer with Applications to Orbits, Aerospace, and Virtual Reality*. Princeton University Press, 1999.

[85] E. B. Dam, M. Koch, and M. Lillholm. "Quaternions, Interpolation and Animation". In: *Daimi PB* 537 (2001).

[86] R. N. Bracewell. *The Fourier Transform and its Applications*. McGraw-Hill, 1986.

[87] R. J. Mailloux. *Phased Array Antenna Handbook, Third Edition*. Artech House, 2017.

[88] M. A. Richards. "Fundamentals of radar signal processing". In: *Fundamentals of radar signal processing* (2005).

[89] W. B. Johnson. "Introduction to Radar Systems". In: *IEEE Aerospace and Electronic Systems Magazine* 28.3 (2013), pp. 4–10.