A close-up of a sign

Description automatically generated

Utilizing RNNs in 4D Radar Point Clouds for Enhanced Autonomous Vehicle Perception

Nick Timaskovs K00260158

# Abstract

This thesis explores the multifaceted realm of object detection, tracking, and classification within the dynamic landscape of 4D radar data in autonomous vehicles. The advent of 4D radar technology has opened the way for a new era of high-resolution, real-time data acquisition, enabling an unprecedented understanding of the complex dynamics of objects in motion within a given space and time. This research delves into the development and refinement of algorithms and techniques designed to enhance the accuracy and efficiency of object detection in 4D radar data. By harnessing the power of deep learning models, recurrent neural networks, and advanced signal processing techniques, this study aims to address the challenges posed by noisy and cluttered radar data, while also improving the speed and precision of object detection.

In addition to object detection, this thesis investigates the critical aspect of object tracking in 4D radar data. Tracking objects through space and time is a fundamental task for various applications, including autonomous vehicles, surveillance, and air traffic control. The research proposes novel approaches that leverage both historical and real-time radar data to robustly track objects, accounting for unpredictable movements, occlusions, and changing environmental conditions. Furthermore, this study expands the scope by incorporating classification into the framework, enabling the automatic identification of objects based on their radar signatures. By integrating these elements into a unified system, this thesis contributes to the advancement of 4D radar technology, opening new possibilities for applications in various domains where precise object detection, tracking, and classification is paramount in ensuring safety in autonomous vehicles.

Table of Contents

[Abstract 2](#_Toc161305277)

[Table of Contents 3](#_Toc161305278)

[Glossary 5](#_Toc161305279)

[Acknowledgements 6](#_Toc161305280)

[2. Literature Review 7](#_Toc161305281)

[2.1 Autonomous Vehicles 7](#_Toc161305282)

[2.2.1 Autonomous Vehicle Perception 8](#_Toc161305283)

[2.3 Overview of Traditional Radar 10](#_Toc161305284)

[2.4 FMCW Radar & Signal Processing 11](#_Toc161305285)

[2.5 Overview of 4D mm Radar 13](#_Toc161305286)

[2.5.1 4D Radar Advantages & Disadvantages 14](#_Toc161305287)

[2.6 Deep Learning 16](#_Toc161305288)

[2.6.1 Convolutional Neural Networks (CNNs) 16](#_Toc161305289)

[2.6.2 Recurrent Neural Networks (RNNs) 18](#_Toc161305290)

[2.6.2 Region-Based Convolutional Neural Network (R-CNN) 19](#_Toc161305291)

[2.7 Object Detection & Tracking 20](#_Toc161305292)

[2.7.1 Traditional Object Detection Techniques 20](#_Toc161305293)

[2.7.2 Radar Object Detection, Tracking & Classification Experiments 22](#_Toc161305294)

[2.8 Object Classification 23](#_Toc161305295)

[2.9 Conclusion 24](#_Toc161305296)

[3.0 Analysis & Design 25](#_Toc161305297)

[3.1 Analysis 25](#_Toc161305298)

[3.1.1 Data Sparsity and Quality 25](#_Toc161305299)

[3.1.2 Dataset Acquisition 25](#_Toc161305300)

[3.1.3 Computational Complexity 26](#_Toc161305301)

[3.1.4 Steep Learning Curve 26](#_Toc161305302)

[3.1.5 Scope 27](#_Toc161305303)

[3.2 Requirements Analysis 28](#_Toc161305304)

[3.2 Dataset 29](#_Toc161305305)

[3.2.1 View-Of-Delft Sensor Setup 30](#_Toc161305306)

[3.2.2 Dataset Frame Information 30](#_Toc161305307)

[3.2.3 Dataset Label Information 31](#_Toc161305308)

[3.3 Data Visualization and Exploration 31](#_Toc161305309)

[3.4 Proposed Object Detection Architecture 32](#_Toc161305310)

[3.5 Overview of PV-RCNN Design 32](#_Toc161305311)

[3.5 PV-RCNN in depth 34](#_Toc161305312)

[3.5.1 3D Voxel CNN For Feature Encoding and Proposal Generation 35](#_Toc161305313)

[3.5.2 Voxel-to-key point Scene Encoding via Voxel Set Abstraction 35](#_Toc161305314)

[3.5.3 Keypoint-to-grid RoI Feature Abstraction for Proposal Refinement 36](#_Toc161305315)

[3.6 Training and Inference Details 38](#_Toc161305316)

[3.7 Training losses 38](#_Toc161305317)

[3.8 Evaluation Metrics 39](#_Toc161305318)

[4.0 Implementation 41](#_Toc161305319)

[4.1 Introduction 41](#_Toc161305320)

[4.2 Development Process 41](#_Toc161305321)

[4.3 Programming Languages 42](#_Toc161305322)

[4.4 Integrated Development Environment (IDEs) 42](#_Toc161305323)

[4.5 Libraries 43](#_Toc161305324)

[4.6 Other Tools 44](#_Toc161305325)

[4.7 Point Cloud Visualization in Open3d 45](#_Toc161305326)

[4.8 Data Loader & Split 46](#_Toc161305327)

[Appendices 49](#_Toc161305328)

# Glossary

ADAS Advanced Driver Assistance Systems

AV Autonomous Vehicle

AI Artificial Intelligence

CW Continuous Wave

RNN Recurrent Neural Network

ML Machine Learning

LSTM Long Short-Term Memory Networks

AoA Angle Of-Arrival

FoV Field Of View

FPS Frames Per Second

GPU Graphics Processing Unit

MIMO Multiple Input Multiple Output

RAD Range-Angle-Doppler

RoI Region Of Interest

RAED Range-Azimuth-Elevation-Doppler

RCS Radar Cross Section

RD Range-Doppler

RADAR Radio Detection and Ranging

Rx Receiver antenna

LiDAR Light Detection and Ranging

Tx Transmitter antenna

SORT Simple and Online Real-time Tracking

FMCW Frequency-Modulated Continuous Wave

VOD View-Of-Delft (Dataset)

# Acknowledgements

I would like to thank my supervisor William Ward for all the support over the course of my final year.

I would like to thank Iurii Ivanov, Dane Mitrev and Clement from Provizio for the support and assistance during the development of my thesis.

Lastly, I would like to thank Barry Lunn the CEO of Provizio for opportunity to be a partner of this thesis and helping this vision come to life.

## 2. Literature Review

## 2.1 Autonomous Vehicles

Autonomous vehicles (AVs) are self-driving vehicles that are equipped with a multitude of sensors illustrated by **Figure 1**, including LiDAR, radar, cameras, ultrasonic sensors, and GPS. These sensors provide information about the vehicle's surroundings, capturing data about other vehicles, pedestrians, road infrastructure, and weather conditions. AVs and ADAS also use advanced AI and ML algorithms to process sensor data to make real-time decisions and interact with vehicle Control Systems such as steering, acceleration, and braking. (Synopsys.com, 2023)

A car with different colored circles

Description automatically generated with medium confidence

**Figure 1**: Sensor setup of the Valeo Drive4U prototype. (Valeo 2018)

One of the key drivers of autonomous vehicles is safety. Self-driving cars have the potential to significantly reduce the number of accidents caused by human error, which accounts for most traffic fatalities. According to the National Highway Traffic Safety Administration (NHTSA), 94% of all traffic accidents are attributed to human error. (TRAFFIC SAFETY FACTS, 2017).

Efficiency is another crucial benefit of autonomous vehicles. They can optimize routes, reduce traffic congestion, and improve fuel efficiency, leading to a more sustainable and eco-friendly transportation system. Moreover, they offer new possibilities for mobility and accessibility for individuals who may have difficulty driving, such as the elderly and disabled.

As this technology advances, it raises important questions regarding regulation, ethics, and cybersecurity. Governments and industries worldwide are actively working on guidelines and standards for autonomous vehicles. While we're not yet in an era where self-driving cars dominate the streets, we are making steady progress towards that future. (NHTSA, 2020)

## 2.2.1 Autonomous Vehicle Perception

Perception is a fundamental component of AVs and advanced driver assistance systems (ADAS), referring to how a vehicle senses or “sees”, enabling them to understand their environment, including identifying and tracking objects, pedestrians, other vehicles, and road infrastructure such as lane markings, traffic signs, and traffic lights. Perception is done with the use of a multitude of sensors such as RADAR, LiDAR, Cameras and Ultrasonic.

A diagram of a car driver level

Description automatically generated with medium confidence**Figure 2:** (2.0 A Vision for Safety AUTOMATED DRIVING SYSTEMS, n.d.)

Illustrated by **Figure 2**, AD and ADAS are categorized into 6 levels, Level 0 (full human control) to Level 5 (complete vehicle autonomy) these levels are defined by the Society of Automotive Engineers (SAE) in their J3016 standard, which provide a framework to understand the progression of autonomy in vehicles.

As ADAS research, testing, and implementation in vehicles continue to grow worldwide, there is an increasing focus on establishing standardized rules and regulations to guarantee their secure incorporation into society. Most commercial vehicles are categorized as Level 1 to Level 2 autonomy due to limitations in sensors, cost factors, and the need for ongoing driver attention and control. These vehicles generally come equipped with ADAS features such as emergency braking, blind spot detection, Adaptive Cruise Control, Automatic Parking, and Lane Assist. (2.0 A Vision for Safety AUTOMATED DRIVING SYSTEMS, n.d.)

At CES 2023, Mercedes-Benz made a significant announcement, stating that it had achieved Level 3 (L3) autonomous driving certification in the United States, specifically from the state of Nevada.

It's worth noting that L3 certification is granted at the state level in the US, so Mercedes' system is only considered L3 in Nevada for now. This move by Mercedes is expected to encourage other major automotive manufacturers such as Hyundai-Kia, Stellantis, BMW, GM, and Honda to pursue Level 3 autonomous driving technology, as they have also been reporting progress and plans for L3 rollout. (AUTOCRYPT, 2023)

SLAM algorithms are employed in autonomous vehicles, they enable the vehicle to create a map of its surroundings while determining its own position within that map simultaneously. SLAM algorithms empower the vehicle to chart unexplored environments, and engineers utilize this map data for tasks like planning routes and avoiding obstacles. (Mathworks.com, 2023)

AVs use sensor fusion techniques to combine data from multiple sensors, improving the accuracy and robustness of perception systems. Kalman filters and Bayesian approaches are commonly employed for sensor fusion. (Anwesh Marwade, 2020)

## 2.3 Overview of Traditional Radar

Radar technology has become the basis for modern sensing and surveillance systems for several decades. Its applications span from military, aviation and maritime navigation to weather monitoring and autonomous vehicles. Radar technology has its roots in the early 20th century, with significant advancements occurring during World War II. Early radar systems used microwave signals to detect and locate objects. Notable developments include the British Chain Home system and the American SCR-270 radar. These systems provided critical advantages in terms of early warning and target tracking. (Bloom, 2020)

A diagram of a signal

Description automatically generated

**Figure 3:** Basic Radar Operation (Roshni Y, 2019)

Illustrated by **Figure 3**, Radar operates on the principles of emitting electromagnetic waves, using a transmitter antenna (Tx) and receiving their echoes via a receiver antenna (Rx) after reflecting off objects and analyzing the time delay and **Doppler shift** of these echoes to determine an object's range, speed, and direction. Key components include transmitters, antennas, receivers, and signal processing units.

**Doppler Shift Definition**: “*the apparent difference between the frequency at which sound or light waves leave a source and that at which they reach an observer, caused by the relative motion of the observer and the wave source*”. (Doppler effect | Definition, Example, & Facts| Britannica, 2023)

## 2.4 FMCW Radar & Signal Processing

There are many different types of RADAR, all used in various use cases as discussed previously however mentioned in **section [2.3]**, FMCW is the most used in automotive. and is relevant to this use case.

FMCW operates using linear frequency-modulated continuous-wave signals to measure range, angle, and velocity. Based on regulations they can use two frequency bands: 24 GHz and 77 GHz, with a preference for 77 GHz due to its wider bandwidth, higher Doppler resolution, and smaller antennas great for long-range use cases. (Udemy, 2023)

FMCW signals have key parameters as shown in **Figure 4:** start frequency (𝑓𝑐), sweep bandwidth (B), chirp duration (𝑇𝑐), and slope (S). A frame consists of chirps with a frame time of (𝑇𝑓). A frequency mixer combines received and transmitted signals to produce two signals: sum frequency 𝑓𝑇(𝑡)+𝑓𝑅(𝑡) and difference frequency 𝑓𝑇(𝑡)−𝑓𝑅(𝑡). with a low-pass filter used to obtain the intermediate frequency (IF) signal. Complex exponential IF signals are achieved in practice using a quadrature mixer. (Zhou et al., 2022)

Range and Doppler velocity can be estimated from the IF signal, leading to a 2D complex data matrix called the Range-Doppler (RD) map. The angle information is obtained using multiple receivers or transmit channels. (Zhou et al., 2022)

A diagram of a graph

Description automatically generated with medium confidence

**Figure 4:** Radar Tx/Rx signals and the resulting range-Doppler map (Zhou and Yue, 2022)

Angle estimation in radar can be achieved using SIMO radar, which involves a single transmitter antenna and multiple receive antennas. By measuring the phase change between adjacent receive antennas, the direction of an object can be calculated using the formula Δ𝜙=2𝜋𝑑sin𝜃/𝜆, where 𝜃 represents the object's angle, and 𝑑 is the antenna spacing. (Zhou et al., 2022)

To achieve maximum angle precision, the antenna spacing can be set to 𝜆/2, and a third Fast Fourier Transform (FFT) is used for processing.

The angular resolution of a SIMO radar depends on the number of receive antennas, but this number is limited by the cost of signal processing.

MIMO radar employs multiple transmit and receive channels, creating a virtual array with many channels. Various techniques like time-division multiplexing (TDM), frequency-division multiplexing (FDM), and Doppler-division multiplexing (DDM) are used to ensure signal separation.

TDM is straightforward, where different transmit antennas take turns sending signals. DDM sends all signals simultaneously but reduces Doppler velocity resolution.

Once signals are separated, a 3D tensor, called the RAD tensor, is created by stacking Range-Doppler (RD) maps. Angular resolution can be enhanced using super-resolution techniques like Capon, MUSIC, and ESPRIT.

The radar detection process involves coherent integration to improve signal-to-noise ratio (SNR), followed by a constant false alarm rate (CFAR) detector for peak detection. Angle estimation is then applied, resulting in a point cloud with range, Doppler, and angle measurements.

In conventional radars, only azimuth angles are resolved, while 4D radars output both azimuth and elevation angles.

Low CFAR thresholds are used in safety-critical applications for high recall. Spatial-temporal filtering techniques, such as DBSCAN and Kalman filtering, are employed to reduce errors caused by clutter and interference.

## 2.5 Overview of 4D mm Radar

4D-imaging radar represents a cutting-edge iteration of mm Wave radar technology that surpasses conventional radar's capabilities.It deploys echolocation and utilizes time-of-flight measurement principles to create a representation of objects within a 3D setting. The four dimensions include **Range, Azimuth, Elevation, and Doppler velocity.** It also provides some other low-level capabilities such as radar-cross-section (RCS) or signal-to-noise ratio (SNR). (Everythingrf.com, 2021)

To create a detailed representation of the surrounding environment for a vehicle, a 4D imaging radar employs a Multiple Input Multiple Output (MIMO) antenna array. This array can consist of numerous antennas that transmit signals towards objects in the vicinity of the device and subsequently collect the reflected signals. The information gathered by these antennas is then utilized to construct a point cloud, depicting the region encompassing the vehicle in high detail. This point cloud captures not only the location and movement of objects but also their shape and texture, allowing for an unprecedented level of environmental awareness. By analysing the Doppler shifts in the returned signals, the system can determine the velocity of moving objects, adding a dynamic aspect to the static scene captured by traditional radar systems. The use of a MIMO antenna array significantly enhances the radar's resolution and accuracy, enabling the detection of smaller objects and finer details in complex urban and highway environments. Consequently, 4D imaging radar plays a crucial role in enhancing the safety and efficiency of autonomous driving systems, offering a comprehensive understanding of the vehicle's immediate surroundings and potential hazards.

A collage of a person walking on a street

Description automatically generated

**Figure 5**: Examples of 4D RADAR objects detected projected to the image plane. (Zhou et al., 2020)

## 2.5.1 4D Radar Advantages & Disadvantages

**Advantages**

Unlike RGB cameras, which utilize the visible light spectrum (384-769 terahertz), and Lidar systems, which operate in the infrared spectrum (361-331 terahertz), Radars operate at significantly longer radio wavelengths (77-81 gigahertz). This characteristic allows Radars to provide reliable measurements (Cohen, 2023) even in adverse weather conditions such as rain, fog, or snow, ensuring reliable performance in various environments. (Yang et al., n.d.)

Radar operates independently of ambient lighting conditions, making it ideal for 24/7 surveillance and autonomous driving at night (Zhou et al., 2020)

By breaking down each azimuth into an array of intensity values distributed radially, radar introduces an additional dimension that Lidar lacks. This unique feature enables radar to construct a top-down, image resembling a photograph, a task that a Lidar unit cannot accomplish without incorporating multiple channels. (Navtech Radar, 2023)

4D Radar offers notable advantages in terms of long-range detection capabilities, with a range extending beyond 200 meters, as well as robustness. Millimetre-wave radar can penetrate certain non-metallic obstacles, such as plastic and fabric allow for radar to be seamlessly hidden behind a bumper for an aesthetic look. (Xx, Xx and Xxxx, n.d.)

**Disadvantages**

By integrating data from various sensors, such as visual sensors and LIDAR, the system can enhance its understanding of the driving environment. Deep learning (DL) techniques have played a pivotal role in this, with a multitude of advanced deep neural networks (DNNs) being employed for perception tasks. Thanks to the substantial learning capacity of DL, there has been a significant improvement in the performance of these tasks. Many DL frameworks have been explored for processing both image and LIDAR data, as they provide ample data for training and validating deep neural networks. In contrast, research on radar-related DL studies has been limited, primarily due to the sparsity of radar data. (Zhou et al., 2020). For more research to be completed using RADAR more data is required.

LiDAR excels in providing higher accuracy by being able to output over 100,000 points per frame while 3D radar outputs only 1,000 points per frame. (Hesai Webmaster, 2023)

Illustrated by **figure 6**, certain objects that are clearly visible in LiDAR data, such as cars and pedestrians, may appear blurred or less well-defined when observed through Radar data. (AR, 2021)

A comparison of a heat map

Description automatically generated with medium confidence

**Figure 6:** Lidar VS Radar Point Cloud (AR, 2021)

A diagram of a blue and orange star

Description automatically generated with medium confidenceTo conclude **Figure 7:** summarises some of the advantages of RADR over LiDAR, however because of Radars limited resolution and the absence of semantic features, radar-based technologies “*for detection and tracking, vehicle self-localization, and HD map updating currently”* not as advanced as other perception sensors in highly autonomous driving. Nevertheless, research efforts in radar technology have been on the rise, thanks to the unique advantages offered by radar sensors. Enhancing the quality and imaging capabilities of millimetre-wave (MMW) radar data, along with exploring the full potential of radar sensors, is essential for gaining a comprehensive understanding of the driving environment. (Zhou et al., 2020)

**Figure 7:** Radar VS Radar Graph (Barnard, 2016)

## 2.6 Deep Learning

Deep learning models consist of a series of interconnected layers, like the human brain that consist of millions of interconnected neurons that cooperate to acquire knowledge. (Gillis, Burns and Brush, 2023) these layers create a continuous and trainable model using backpropagation. Researchers have extensively studied these layers and neural network structures in recent years, leading to enhancements in data feature representations. The utilization of extensive annotated datasets has enabled the training of models with larger parameter counts, resulting in improved performance on public benchmarks and challenges.

Deep Learning applications employ a hierarchical set of algorithms referred to as an artificial neural network (ANN). The architecture of such an ANN is inspired by the biological neural networks found in the human brain, enabling a learning process that surpasses the capabilities of conventional machine learning models. (Levity.ai, 2023)

This section will focus on the state-of-the-art deep learning models for computer vision tasks such as object detection, tracking and classification.

## 2.6.1 Convolutional Neural Networks (CNNs)

Images captured by cameras are represented as 3D matrices with pixel values ranging from 0 to 255 for each of the Red, Green, and Blue (RGB) channels. When utilizing images as input for a machine learning model, it becomes computationally intensive. For instance, if we consider an image with dimensions 200x200x3, a single fully connected layer requires 120,000 parameters to process each pixel. This conventional ANN cannot efficiently handle high-dimensional input data and cannot directly learn spatial features from the input. (Ouaknine, 2022)

In recent years, Convolutional Neural Networks (CNNs) have been explored to address these challenges. CNNs utilize Convolutional layers are comprised of a grid of neurons, with a requirement that the previous layer also consists of a grid of neurons in a rectangular shape. Each neuron in this layer receives inputs from a rectangular segment of the preceding layer, with the same set of weights applied to this rectangular segment for all neurons within the convolutional layer. Consequently, the convolutional layer essentially performs an image convolution operation on the previous layer, using these weight values to define the convolution filter. (Joel Markus Vaz and Balaji, 2021)

Within each convolutional layer, there can be multiple grids, each of which obtain inputs from all the grids in the preceding layer, utilizing potentially distinct filters.

Following each convolutional layer, there is a pooling layer illustrated by **Figure 8**.

A diagram of a diagram of a process

Description automatically generated with medium confidence

**Figure: 8** Diagram of a CNN (ResearchGate, 2019)

The pooling layer selects small rectangular blocks from the convolutional layer and reduces them to a single output from that block through subsampling. Various pooling methods can be employed, such as averaging, taking the maximum value, or using a learned linear combination of the neurons within the block. In this context, our pooling layers consistently utilize max pooling, which means they pick the maximum value from the block they are pooling. (Joel Markus Vaz and Balaji, 2021)

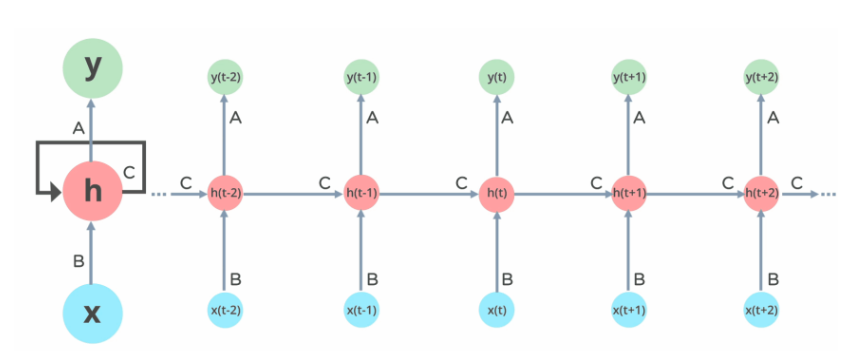
Finally, after a series of convolutional and max-pooling layers, the neural network's high-level reasoning occurs through fully connected layers. A fully connected layer connects all the neurons from the previous layer, whether it was a fully connected, pooling, or convolutional layer, to every neuron within itself. Fully connected layers do not have a spatial arrangement (they can be envisioned as one-dimensional), making it infeasible to introduce convolutional layers after a fully connected layer. (Gibiansky, 2014)

## 2.6.2 Recurrent Neural Networks (RNNs)

Recurrent neural networks (RNNs), like other deep learning methods, have been around since the 1980s, but their true capabilities became evident only recently. The introduction of long short-term memory (LSTM) in the 1990s, along with greater computing power and the abundance of data, has propelled RNNs to the forefront of machine learning. (Kalita, 2022)

An RNN is a neural network designed for handling sequential data, and it finds application in various fields, including temporal series analysis, such as music, video, and stock market data, as well as Natural Language Processing tasks like textual analysis and translation.

In a RNN the output from the previous step serves as input to the current step. Unlike traditional neural networks, where inputs and outputs are treated independently. (GeeksforGeeks, 2018)



**Figure: 9** RNN Architecture (Avijeet Biswal, 2020)

Illustrated by **Figure: 9** "x" represents the input layer, "h" corresponds to the hidden layer, and "y" denotes the output layer. The network's performance is enhanced by utilizing parameters A, B, and C. At each time step "t," the present input is a composite of both the current input at x(t) and the previous input at x(t-1). The output at any given time is looped back into the network to refine the model's output. (Avijeet Biswal, 2020)

Recurrent Neural Network (RNN) models offer several advantages, such as their ability to handle sequences of varying lengths, their consistent number of parameters regardless of input size, and their capacity to incorporate historical context by sharing parameters across timestamps. Nevertheless, RNNs are hindered by their slow training speed and the issue of vanishing gradients, which tend to occur towards the end of sequences. Additionally, these models struggle to effectively capture long-term dependencies due to information loss during sequence processing. (Ouaknine, 2022)

## 2.6.2 Region-Based Convolutional Neural Network (R-CNN)

This approach begins by employing a selective search technique Initially, it segments an image into smaller regions and then merges them hierarchically using various colour spaces and similarity metrics (Jasper et al., 2013). This process yields a limited set of region proposals that may potentially contain objects of interest. The R-CNN model, proposed by (Girshick et al., 2016), combines this selective search method for region proposal generation with deep learning for object classification illustrated by **Figure 10.**

A diagram of a person

Description automatically generated

**Figure: 10** Diagram of a RCNN (Gandhi, 2018)

Each region proposal is resized to match the input requirements of a Convolutional Neural Network (CNN), which produces a 4096-dimensional feature vector. This feature vector is then used as input for binary Support Vector Machine (SVM) classifiers, one for each class. Additionally, it is used in a linear regressor to adapt the shapes of the corresponding bounding boxes, reducing location errors. (Hearst et al., 1998)

The CNN is trained on the 2012 ImageNet dataset for image classification and then fine-tuned using region proposals that have an Intersection over Union (IoU) greater than 0.5 with the ground-truth bounding boxes. Two versions of the model are created: one using the 2012 PASCAL VOC dataset and the other using the 2013 ImageNet dataset with associated bounding boxes. SVM classifiers are also trained for each class within both datasets. The top-performing R-CNN models have achieved a mean Average Precision (mAP) score of 62.4% in the 2012 PASCAL VOC challenge, representing a substantial 22.0-point improvement over the second-best result on the leaderboard. Furthermore, they achieved a 31.4% mAP score on the 2013 ImageNet dataset, surpassing the second-best result on the leaderboard by 7.1 points. (Ouaknine, 2022)

## 2.7 Object Detection & Tracking

Object detection represents a fundamental task in the realm of computer vision and holds significant importance in enabling autonomous driving. Autonomous vehicles heavily rely on their ability to perceive their surroundings, ensuring safe driving performance. To achieve this, they utilize object detection algorithms, which accurately identify objects such as pedestrians, vehicles, traffic signs, and barriers within their proximity. Deep learning-based object detectors are instrumental in real-time identification and localization of these objects. (Balasubramaniam and Pasricha, n.d.)

This section explores the current state of the art in object detection and highlights the open challenges associated with integrating these technologies into autonomous vehicles.

## 2.7.1 Traditional Object Detection Techniques

The modern progression of object detection techniques initiated two decades ago with the Viola Jones detector, in 2001, Paul Viola and Michael Jones introduced an object recognition framework for real-time human face detection. This framework employs sliding windows to scan an image at various locations and scales to identify human faces. The search is based on "haar-like" features, named after Alfred Haar, who pioneered the concept of haar wavelets. These wavelets serve as the image's feature representation. To expedite detection, an integral image is utilized, ensuring that the computational effort for each sliding window remains independent of its size. (baeldung, 2022).

The authors also employ the Adaboost algorithm for feature selection, which identifies a small set of features that are particularly useful for face detection from a large pool of random features. Additionally, the algorithm incorporates Detection Cascades, a multi-stage detection approach aimed at reducing computational overhead. This means that it prioritizes less computation on background windows and focuses more on potential face targets. (Borah, 2020)

A few years later, the Histogram of Oriented Gradient (HOG) detectors gained popularity, especially for detecting pedestrians (Tyagi, 2021). These HOG detectors were subsequently expanded into Deformable Part-based Models (DPMs), representing the first models geared towards the detection of multiple objects. (Davies, 2022)

Around 2014, the surge in interest surrounding deep neural networks led to a significant breakthrough in multiple object detection with the introduction of the Regions with Convolutional Neural Network (R-CNN) deep neural network model. This innovation resulted in a remarkable 95.84% enhancement in Mean Average Precision (mAP) over the existing state-of-the-art methods.

This pivotal advancement not only redefined the effectiveness of object detectors but also made them appealing for entirely new application domains, particularly in the context of Autonomous Vehicles (AVs).

Since 2014, the continued evolution of deep neural networks and the progress in Graphics Processing Unit (GPU) technology have paved the way for faster and more efficient object detection in real-time images and videos. Modern AVs heavily rely on these improved object detectors for various crucial tasks, including perception, path planning, and other decision-making processes.

**A diagram of a device

Description automatically generated**

**Figure 11:** Taxonomy of object detectors (Balasubramaniam and Pasricha, 2022)

## 2.7.2 Radar Object Detection, Tracking & Classification Experiments

To emphasize the importance of the 4DRT-based perception module, (Paperswithcode.com, 2022) introduced a basic neural network for 3D object detection (referred to as the baseline NN) that takes 4DRT as its input. Their experiments on the K-Radar dataset reveal that the 4DRT-based baseline NN excels in the task of 3D object detection, particularly in challenging weather conditions, outperforming the Lidar-based network.

In another study carried out by (Scheiner et al., 2021), the objective is to perform object detection on automotive radar point clouds to identify moving road users using two end-to-end object detectors (YOLOv3 and PointPillars). YOLOv3 performs the best with a mean Average Precision (mAP) of 53.96%, offering potential for combining static and dynamic object detection in the future.

PointPillars, a point-cloud-based object detector, lags behind with a 36.89% mAP. Improving its performance to 45.82% by using an enhanced CNN backbone still falls short of YOLOv3-like results. While point-cloud-based detectors have potential, they are not yet on par with image-based variants. However, ongoing model development may bridge this performance gap, offering increased flexibility for various sensor types and improved speed in the future.

Another study comparing SSD and Faster R-CNN on Radar Imagery done by (Stroescu et al., 2021), Concluded the training loss for SSD consistently decreases with each training epoch, the Faster R-CNN loss exhibits significantly lower values, leading to higher mean Average Precisions (mAPs). One plausible reason for the superior performance of Faster R-CNN can be learned from a study done by (Huang et al., 2016), which delves into the trade-off between speed and accuracy among various detectors on the COCO dataset. This research found that Faster R-CNN excels in accuracy, whereas SSD, though faster, struggles significantly when it comes to detecting small objects.

## 2.8 Object Classification

One of the widely pursued objectives in the field of computer vision involves the task of classification, which entails assigning a specific category to every image within a dataset. The ImageNet dataset, introduced by (Deng et al., 2009), has played a significant role in extensive exploration and research on classification. A vector is used to measure and represent the local characteristics of an image, summarizing these features in the form of a histogram that describes the overall image. This section will provide an overview of the evolution and improvements made over the years in image classification.

**AlexNet (2012):** A deep convolutional neural network known as AlexNet, created by Alex Krizhevsky and his team in 2012, emerged victorious in the ImageNet Large Scale Visual Recognition Challenge (ILSVRC). This groundbreaking architecture featured numerous convolutional layers and served as a pioneering example of how deep learning could be harnessed for the purpose of image classification. (Nitin Kushwaha, 2023)

**VGGNet (2014):** The VGGNet design, created by the Visual Geometry Group at the University of Oxford, focused on creating deeper neural network architectures. Its straightforward and consistent structure set a standard for investigating network depth. (Deepchecks, 2021)

**ResNet (2015):** Residual Networks, commonly known as ResNets, revolutionized deep neural networks by introducing the concept of residual connections, effectively tackling the issue of vanishing gradients. This breakthrough enabled the successful training of extremely deep networks, leading to the practical realization of ResNets with hundreds of layers, resulting in significant performance enhancements. They achieved remarkable success by securing the top position in the ILSVRC 2015 classification competition with a top-5 error rate of 3.57%, using an ensemble model. (Great Learning Team, 2020)

**DenseNet (2017):** DenseNet introduced a novel approach to connectivity patterns by promoting dense connections, which encouraged the reutilization of features and facilitated the flow of gradients. This architectural innovation showcased enhanced training effectiveness and increased accuracy in the context of image classification tasks. (Paperswithcode.com, 2020)

This section discusses the accomplishments in deep learning related to image classification, specifically in the context of the ImageNet challenge. It provides detailed information about well-known modules and architectures that are still widely employed for feature extraction today. It is important to note that this isn't an exhaustive catalogue of all the models developed between 2012 and 2018. Furthermore, the more recent advancements, particularly in Transformer architectures as proposed by (Dosovitskiy et al., 2020) fall outside the scope of this thesis.

## 2.9 Conclusion

## 3.0 Analysis & Design

## 3.1 Analysis

Implementing a Point-Voxel RCNN (PV-RCNN) for object detection from 4D radar point clouds is a very challenging task for a final year thesis and it presents a massive set of unique challenges. These challenges stem from the nature of radar data, the complexity of the PV-RCNN architecture and the lack industry knowledge in such a specific niche of software development for an undergraduate. This analysis aims to highlight these challenges to minimise the risk of running into too many blockers during the implementation phase.

## 3.1.1 Data Sparsity and Quality

Radar point clouds are inherently sparser than those obtained from LiDAR, which can make it difficult to detect and classify objects with high accuracy. Furthermore, radar data can be affected by noise and clutter from the environment, such as reflections from ground surfaces or nearby objects, making it challenging to isolate and identify relevant features for object detection.

## 3.1.2 Dataset Acquisition

Obtaining access to datasets with 4D radar data and ground truth annotations for training and validation can hinder the development and evaluation. These datasets, rich in spatial and velocity information, are crucial for the development and validation of object detection models like PV-RCNN. However, the scarcity of publicly available 4D radar data can be attributed to several factors, including the proprietary nature of the technology and concerns over competitive advantage.

Many companies in the automotive and tech industries invest heavily in sensor technology and data collection, viewing their datasets as valuable assets that provide a competitive edge. As these companies race to develop the best and most innovative technologies, they often restrict access to their data to protect their advancements and intellectual property. This competitive landscape makes it difficult for researchers and smaller entities to access high-quality, diverse 4D radar datasets needed for advancing research and development in object detection and autonomous vehicle technologies. This limitation not only hampers collaborative efforts that could accelerate technological advancements but also poses a barrier to entry for new players and academic researchers striving to contribute to the field.

## 3.1.3 Computational Complexity

The computational complexity of advanced object detection models, such as the PV-RCNN architecture adapted for 4D radar point clouds, necessitates the use of powerful computing resources, typically involving multiple Graphics Processing Units (GPUs). These models perform complex operations, including voxelization, feature extraction, and the integration of point-level and voxel-level information, which are computationally intensive processes.

This limitation can lead to extended training times, reduced time to develop and ultimately may constrain the scope and ambition of the project. The challenge is compounded by the iterative nature of machine learning projects, where multiple experiments and parameter tunings are essential for optimizing model performance. Without the computational resources typically available in well-funded labs or companies.

## 3.1.4 Steep Learning Curve

The challenge of navigating a niche and technically complex task of object detection using 4D radar point clouds is particularly daunting for an undergraduate student, largely due to the steep learning curve and the lack of accessible industry knowledge. This specialized domain not only requires a deep understanding of advanced concepts in radar technology, computer vision, and machine/deep learning but also necessitates insights into the current state of the art and industry practices that are often not available in academic curricula or publicly accessible resources. The proprietary nature of much of the research and development in this area further exacerbates the difficulty in obtaining relevant and up-to-date information. This means that beyond mastering the technical skills, there's an additional layer of challenge in simply understanding the context, applications, and potential limitations of this project.

## 3.1.5 Scope

The project's scope for tackling the task of object detection, tracking and classification will need to be deliberately concentrated on the detection of objects within point clouds, due to the challenges explained in the previous paragraphs and due to time constraints.

This strategic limitation is critical to manage complexity, as it allows for a more manageable development process and provides a strong basis for future expansion into tracking and classification once detection is mastered.

## 3.2 Requirements Analysis

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Requirement ID | Requirement Type | Description | Priority | Verification Method |
| R1 | Functional | The system must be able to detect objects in 3D point cloud data. | High | Test with standard datasets |
| R2 | Functional | The system should implement the PV-RCNN algorithm for object detection. | High | Code review and execution |
| R3 | Functional | The system must classify detected objects into predefined categories. | Medium | Test with labelled datasets |
| R4 | Performance | The object detection model should achieve a minimum average precision (AP) of X% on a standard validation dataset. | High | Quantitative evaluation |
| R5 | Performance | The system should process point cloud data and return results within Y seconds per frame. | Medium | Timing tests |
| R6 | System | The system must be compatible with GPU acceleration for model training and inference. | High | Hardware compatibility check |
| R7 | System | The system should support input from common 3D point cloud data formats (e.g., PCD, PTS). | Low | Functional testing with different file formats |
| R8 | Functional | The system should provide a user interface for visualizing detection results. | Low | User testing |
| R9 | Performance | The system should maintain robust detection performance under varying environmental conditions (e.g., weather, lighting). | Medium | Test under different simulated conditions |
| R10 | System | The system should be documented, including installation, usage, and troubleshooting guidelines. | Medium | Review of documentation completeness |
| Requirement ID | Requirement Type | Description | Priority | Verification Method |
| R1 | Functional | The system must be able to detect objects in 3D point cloud data. | High | Test with standard datasets |
|  |  |  |  |  |

## 3.2 Dataset

Several datasets with 4D radar have been released and applied in recent years. However, despite the lack of publicly available datasets, an analysis was conducted on the datasets that where available. From the analysis the author created the following summary:

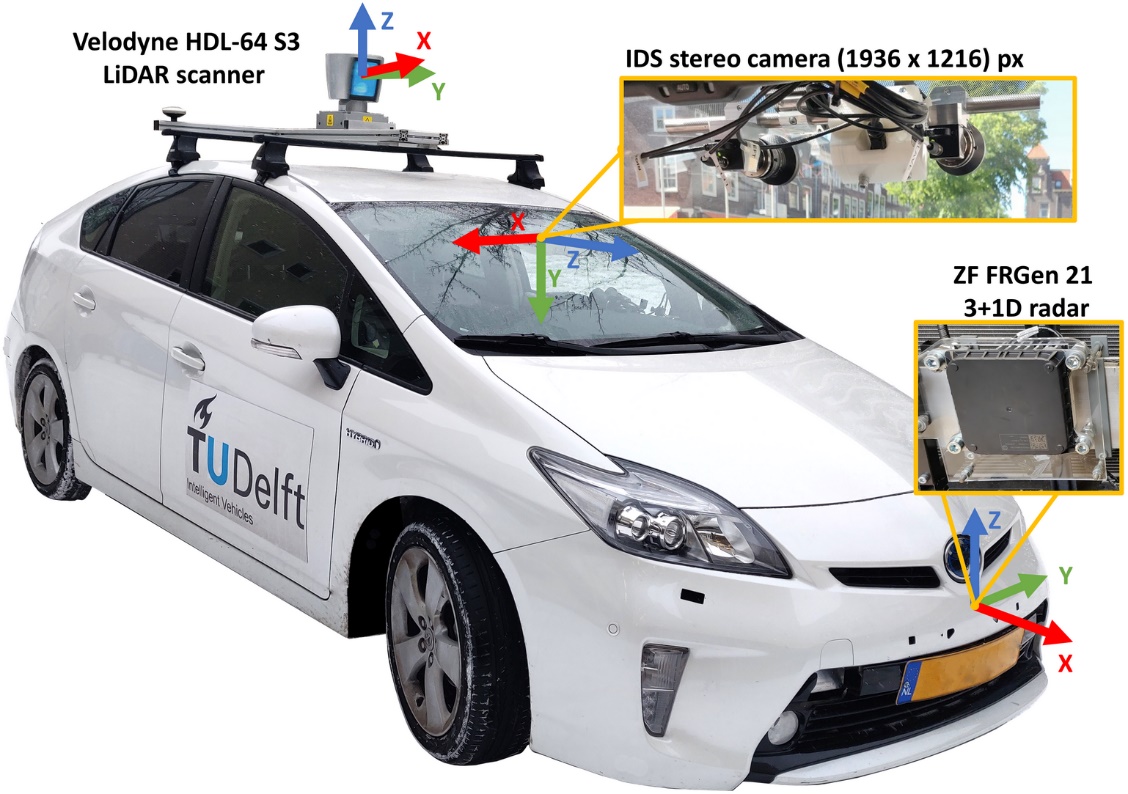
Astyx, an early-released dataset, provides rich data for 3D object detection but is limited by its small size of 546 frames and 3000 object annotations, lacking special scenarios and urban data. RADIal offers a medium-scale dataset with urban streets and highways but lacks 3D bounding boxes and tracking IDs and does not cover adverse weather conditions. View-of-Delft addresses the object tracking problems present in the other datasets with 8,693 frames and 120,000 annotated objects but has a short detection range and lacks 4D radar information for long-range mode. TJ4DRaDSet includes various driving scenarios but lacks data in middle and short-range modes and scenarios with adverse weather conditions. K-Radar provides rich driving scenarios with adverse weather conditions but lacks 4D radar point cloud in long-range mode.

View-of-delft was chosen for this experiment setup as it “consists of more than 123000 3D bounding box annotations, including more than 26000 pedestrian, 10000 cyclist and 26000 car labels.” (GitHub, 2024) Which is great for an object detection problem.

The View-of-delft researchers have an open-source GitHub repository containing useful guides to their dataset along with visualisation and evaluation scripts they used when conducting their research, this repository would come in very useful when exploring the dataset and getting familiar with the process of doing object detection with 4D radar point clouds.

## 3.2.1 View-Of-Delft Sensor Setup

The radar sensor utilized is a ZF FRGen21 3+1D radar, operating at approximately 13 Hz, and is mounted behind the vehicle's front bumper. The provided radar point clouds undergo ego-motion compensation to account for any motion between radar and camera data capture, ensuring consistency when overlaying both datasets. (GitHub, 2024)



**Figure 12:** Prius Sensor Setup VOD (GitHub, 2024)

## 3.2.2 Dataset Frame Information

These radar point clouds are stored in bin files, with each file containing a set of points represented as a Nx7 array, where N is the number of points, and 7 denotes the number of features: *[x, y, z, RCS, v\_r, v\_r\_compensated, time].* Here, *"v\_r”* signifies the relative radial velocity, *“v\_r\_compensated”* indicates the absolute radial velocity compensated for ego motion, and “*time”* denotes the point's time ID, indicating its originating scan. Like Lidar, the provided example by VOD demonstrates data retrieval and plotting. Notably, the 3D plot showcases one significant advantage of radar point clouds over Lidar, as it captures the radial velocity of each measurement alongside spatial information. The labels include the ground truth data for the frame in kitti format. (GitHub, 2024)

## 3.2.3 Dataset Label Information

Sensor fusion in autonomous systems integrates data from multiple sensors like LiDAR and radar to enhance accuracy and robustness, utilizing LiDAR's high-resolution for precise object labelling which is applied to radar data for additional insights such as velocity. Aligning sensor data in both space and time is crucial, often using LiDAR as the benchmark for its detailed spatial data. LiDAR's clear 3D point clouds simplify labelling tasks, with labels easily transferred to radar data for consistency. The dataset's structure prioritizes LiDAR, reflecting its initial design focus and convenience in labelling, leading to radar data being adjunct and reliant on LiDAR for annotations.

For frame number 1047, the first line of the label information could be interpreted as follows:

**Object Type:** rider.

**Truncation**: 1 (possibly indicating high confidence or manual verification)

**Occlusion**: 0 (fully visible)

**Observation Angle**: 1.716500830699201

**Bounding Box**: 979.41486 789.5281 1018.06165 866.89154

**3D Object Dimensions**: 1.503325462332693 (height), 0.7167884312694952 (width), 0.6358283468841199 (length)

**3D Object Location**: 0.7805723338707173 (x), 4.960184749066411 (y), 31.026849236059597 (z)

**Rotation Y**: -4.541531818868102

**Score**: 1 (high confidence or manually verified)

## 3.3 Data Visualization and Exploration

Before any work can be carried out, the radar scans need to be visualised in a point cloud for easier interpretation. Open3D is preferred for its extensive feature set, active community support, cross-platform compatibility, seamless Python integration, and ongoing development. These factors make it a versatile choice for various 3D data processing tasks compared to K3D. Once the point cloud is visualised its good practice to get the annotations from the dataset and attempt to draw the 3D bounding boxes inside of the point cloud to be able to interpret the different objects (car, pedestrian, cyclist etc.) present in the scene.

It would also be beneficial to calculate the number of points within a specific scene.

## 3.4 Proposed Object Detection Architecture

Following an in-depth comparison of object detection architectures, it was decided to use the PV-RCNN architecture from the OpenPCDet library as a guide and reference.

The PV-RCNN architecture combines 3D voxel CNNs and PointNet for precise 3D object detection from point clouds. It leverages voxel CNNs for efficient feature learning and high-quality proposal generation, while incorporating PointNet's ability to capture detailed contextual information. The framework introduces voxel-to-key point scene encoding to condense scene features into key points and point-to-grid RoI feature abstraction for refining proposal confidence and location, effectively integrating the strengths of both network types for enhanced performance.

## 3.5 Overview of PV-RCNN Design

A diagram of a process flow

Description automatically generated

**Fig 13:** OpenPCDet PV-RCNN Architecture Diagram (GitHub, 2024)

Breaking down the architecture into each component based on **Figure 13.**

**1. VFE (Voxel Feature Encoding):** This is the initial stage where raw point clouds are converted into a structured voxel representation. The VFE learns to encode the raw point cloud data into a fixed-size feature representation for each voxel, which can capture the essential information while reducing the sparsity of point clouds.

**2. Backbone3D**

**3D SparseConv:** Once the point clouds are voxelized, the 3D Sparse Convolutional Network processes these voxels to generate high-dimensional sparse feature volumes. It is designed to efficiently handle the sparsity in voxelized point clouds by only computing features at occupied voxels.

**VSA (Voxel Set Abstraction):** The VSA module is responsible for summarizing the voxelized features into a smaller set of key points. This abstraction process leverages the strength of PointNet++ to learn a more discriminative feature representation from the unordered point set within each voxel.

**3. Backbone2D**

**Reshape to BEV (Bird's Eye View):** The high-dimensional features from the 3D backbone are reshaped into a 2D representation corresponding to the bird's eye view of the scene. This process essentially projects the 3D features onto a 2D plane.

**Encoder conv2d:** This 2D feature map is then processed by a 2D convolutional encoder which further refines the features and captures spatial relationships in the bird's eye view.

**4. DenseHead**

**RPN Head (Region Proposal Network):** The RPN Head uses the refined 2D feature maps to generate region proposals. These proposals are candidate regions where objects might be present.

**Point Head:** Parallel to the RPN Head, the Point Head processes the point features from the VSA to generate key point features that describe the objects in the scene.

**5. RoI Head**

**Proposal Layer:** This layer takes the region proposals from the RPN Head and the key point features from the Point Head to generate accurate 3D bounding box proposals.

**RoI-grid Pooling:** The RoI-grid Pooling module aggregates the key point features onto a structured grid within each 3D RoI using set abstraction operations. This step is crucial for capturing the local context around each proposal.

**6. PVRCNN Head:** Finally, the PV-RCNN Head combines the features from the RoI-grid Pooling module with the proposals from the RoI Head to refine the proposals and predict the final bounding boxes along with the object classification.

## 3.5 PV-RCNN in depth

Grasping the intricacies of this architecture necessitates a detailed examination of its critical elements and their interconnected processes. Figure 14 presents a detailed breakdown of the architecture, describing specific functions within each stage for a comprehensive understanding. This section is based on the research carried out by (Shi et al., 2022)

A diagram of a diagram

Description automatically generated

**Fig 14:** More detailed PV-RCNN Architecture Diagram (Shi et al., 2022)

## 3.5.1 3D Voxel CNN For Feature Encoding and Proposal Generation

State-of-the-art 3D detectors commonly employ Voxel CNN with 3D sparse convolution for transforming point clouds into sparse 3D feature volumes due to its effectiveness and precision. The researchers have chosen it as the core of their framework for encoding features and generating 3D proposals. Initially, the input points are segmented into small voxels of size L × W × H, where the attributes of filled voxels are determined by averaging, the features of all points contained within them, typically including 3D coordinates and reflectance values. The network applies multiple 3D sparse convolutions of size 3 × 3 × 3 to progressively convert the point clouds into feature volumes at varying resolutions. For 3D proposal creation, the 8× reduced 3D feature volumes are transformed into,

2D top-view feature maps, from which high-quality 3D proposals are derived using anchor-based methods. This approach, using a 3D voxel CNN backbone, has demonstrated superior recall rates compared to PointNet-based methods. (Shi et al., 2022)

However, these advanced detectors typically use two-stage frameworks that have limitations, such as the reduced spatial resolution of the 3D feature volumes, which can impede precise object localization, and the sparsity of the upscaled feature volumes. To address these challenges, research suggest incorporating a 3D voxel CNN with set abstraction operations to enhance the accuracy and robustness of the second-stage proposal refinement. Despite the potential of set abstraction operations,

directly applying them for pooling scene feature voxels is not practical due to high memory demands and inefficiency. To overcome this, a proposed novel method that first condenses the scene's voxels into a limited set of keypoints at various neural layers, which are then aggregated to RoI grids to refine box proposals efficiently is suggested.

## 3.5.2 Voxel-to-key point Scene Encoding via Voxel Set Abstraction

The approach for encoding 3D scenes from point clouds by transitioning voxel representations into a condensed set of keypoints. This process utilizes a **Voxel Set Abstraction (VSA)** module, which compresses the extensive voxel data from various neural network layers into a manageable number of keypoints. These keypoints are strategically sampled using the **Furthest Point Sampling (FPS)** algorithm to ensure they are evenly distributed and effectively represent the entire scene.

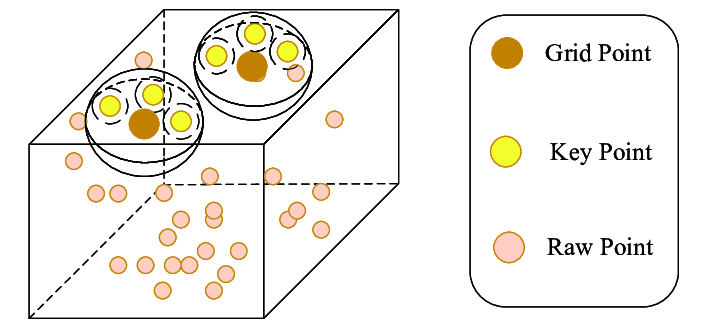
The fundamental concept of FPS involves repeatedly choosing the point that is the greatest distance from the points chosen earlier. This process results in a smaller subset of points that closely represents the original set, maintaining its key characteristics while reducing the total number of points.

The VSA module operates by encoding multi-scale semantic features from the 3D CNN feature volumes into these keypoints. Instead of working with the original, irregularly positioned points, the VSA module aggregates features from the regularized voxels, which have already been processed by the 3D voxel CNN, thereby retaining critical semantic information at multiple scales. This abstraction allows for a more efficient representation of the scene, reducing the computational load for subsequent processing stages, such as proposal refinement. (Shi et al., 2022)

## 3.5.3 Keypoint-to-grid RoI Feature Abstraction for Proposal Refinement

After summarizing the entire scene into a concise set of keypoints enriched with multi-scale semantic features, the framework focuses on refining 3D proposals (Regions of Interest, or RoIs) generated by the 3D voxel CNN. This refinement is achieved by pooling the features from these keypoints to the RoI-grid points using a method called RoI-grid pooling via set abstraction. This method involves sampling grid points within each RoI, identifying neighbouring keypoints for each grid point, and aggregating their features using a PointNet block. This process not only captures rich contextual information from multiple scales but also outperforms previous 3D RoI pooling methods by utilizing flexible receptive fields that extend beyond the RoI boundaries, thus incorporating valuable information from surrounding keypoints.

The RoI-grid pooling module in the PV-RCNN framework enhances the feature aggregation process for 3D object detection proposals illustrated by **Figure 15**. This module operates by systematically mapping the features of a set of keypoints, which encapsulate the scene's essential information, onto a structured grid within each 3D Region of Interest (RoI). Specifically, it begins by uniformly sampling a grid of points, typically in a 6x6x6 arrangement, within each RoI. Each grid point then undergoes a process to identify its neighbouring keypoints based on a predefined radius, ensuring that only relevant features are considered for aggregation.



**Fig 15:** RoI-grid pooling module (Guan, Wan and Jiang, 2023)

The aggregation itself is executed using a PointNet block, which processes the identified neighbouring keypoints to generate a comprehensive feature representation for each grid point. This method allows for the integration of features from multiple scales and receptive fields by setting various radii for neighbourhood keypoint selection. As a result, the aggregated features at each grid point encapsulate a rich context of the surrounding area, extending even beyond the immediate boundaries of the RoI.

This approach contrasts sharply with traditional 3D RoI pooling methods, which often result in the averaging of point-wise features within the RoI or worse, in pooling many uninformative zeros. Instead, the RoI-grid pooling module leverages flexible receptive fields, to gather more meaningful, context-rich information from keypoints around and beyond the RoI, illustrated by **Figure 15**. Once the features for all grid points within an RoI are aggregated, they are vectorized and passed through a Multi-Layer Perceptron (MLP) to create a unified feature representation for the RoI, enhancing the subsequent proposal refinement process.

## 3.6 Training and Inference Details

The PV-RCNN framework from OpenPCDet undergoes training from the ground up in a seamless end-to-end process utilizing the ADAM optimization algorithm. The researchers utilized the KITTI dataset to train the entire network using a batch size of 24 and a learning rate of 0.01 across 80 epochs. This training is performed on 8 GTX 1080 Ti graphics processing units and completes in approximately 5 hours.

Given that the author of this thesis, training setup consists of a laptop with a single Nvidia 1660 Ti GPU, the author may need to consider downsizing the dataset and be prepared for the training duration to extend over several days. This is due to the lower computational power compared to the original setup with 8 GTX 1080 Ti GPUs. Alternatively, to maintain the scale of the dataset and potentially accelerate the training process, the author could leverage a cloud computing service that offers more powerful GPU resources. This would allow me to conduct the training more efficiently, albeit at a potential increase in cost.

## 3.7 Training losses

Based on research and following the methodology of couple papers, the goal is to train the PV-RCNN framework comprehensively from beginning to end using three types of losses:

**Region Proposal Loss (Lrpn):**

The region proposal loss is a combination of classification loss (Lcls) and regression loss. Classification loss is computed using focal loss, a variant of cross-entropy loss that applies a modulating term to focus learning on hard negative examples. This helps in addressing class imbalance during training.

Regression loss is computed using Smooth L1 loss, which is less sensitive to outliers than mean squared error. It is used to predict the offset values for bounding box coordinates (x, y, z) and dimensions (length, height, width) as well as the rotation angle (θ) of the anchor boxes.

The predicted residuals for each anchor box are compared with the ground truth regression targets to compute this part of the loss.

**Keypoint Segmentation Loss (Lseg):**

This loss is also calculated during **section 3.5.2**. The keypoint segmentation loss helps the network to accurately segment the point cloud into objects and background, which is a crucial step for detecting the keypoints that will be used for object proposals.

**Proposal Refinement Loss (Lrcnn):**

The proposal refinement loss consists of the IoU-guided confidence prediction loss (Liou) and the box refinement loss. The IoU (Intersection over Union) loss ensures that the model predictions are closely aligned with the ground truth in terms of overlap, which directly relates to detection accuracy.

Like the region proposal loss, the box refinement loss is computed with Smooth L1 loss. It refines the initial predictions by learning the box residuals with respect to the proposal regression targets, which are encoded similarly to the anchor regression targets.

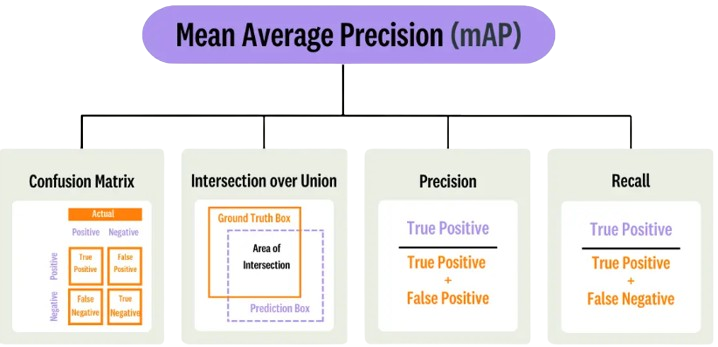
## 3.8 Evaluation Metrics

Based on research the author chose the mean Average Precision (mAP) metric for evaluating the effectiveness of the model.

It utilizes the following subsidiary metrics:

1. Confusion Matrix
2. Intersection over Union (IoU)
3. Precision
4. Recall

It is the average of the Average Precision (AP) for each class. AP for a single class is calculated by plotting a Precision-Recall (P-R) curve, which graphs the precision of the model against its recall at different thresholds. Precision measures the accuracy of the positive predictions, while recall quantifies the model's ability to identify all relevant instances. For each class, AP is computed as the area under the P-R curve, reflecting the model's accuracy across all levels of recall. The mAP is then determined by taking the mean of the AP scores across all classes. This provides a single performance summary that balances both the precision and the ability of the model to detect each class, making it a comprehensive metric for tasks with multiple classes to be detected.



**Fig 13:** mAP Overview (kili-website, 2023)

mAP, or mean Average Precision, is often preferred over other evaluation metrics for object detection tasks due to its comprehensive assessment of both precision and recall across various levels of confidence thresholds. Unlike single-point metrics like accuracy, mAP considers the entire precision-recall curve, providing a more nuanced understanding of a model's performance. This makes mAP particularly suitable for scenarios where a balance between precision and recall is crucial, such as in real-world applications where false positives and false negatives carry different consequences. Additionally, mAP enables fair comparisons between different models by accounting for their performance across all possible operating points, offering a robust measure of overall detection quality.

## 4.0 Implementation

## 4.1 Introduction

This chapter delves into the implementation of the Point-Voxel CNN (PV-RCNN) architecture, tailored for object detection from 4D radar point clouds. The adaptation of PV-RCNN to 4D radar data marks a significant step forward in enhancing the capabilities of autonomous driving systems and advanced driver-assistance systems (ADAS), leveraging the additional velocity information provided by 4D radar to improve detection accuracy and reliability under various environmental conditions.

The implementation process encompasses several key phases, starting with the preprocessing of 4D radar point clouds to make them compatible with the PV-RCNN architecture. This involves the transformation of raw radar data into a structured format that retains spatial and velocity information essential for object detection tasks. Following preprocessing, the author undertakes the task of modifying the PV-RCNN architecture to effectively process and interpret 4D radar data to enhance feature representation and object detection capabilities.

## 4.2 Development Process

The development process for the PV-RCNN architecture, undertaken by the author working in a sole capacity with the support of a third-party partnership, was grounded in the Agile methodology. This approach was particularly suitable given the complexity and the exploratory nature of designing such an advanced neural network. Agile's iterative development cycles, known as sprints, allowed the author to break down the immense task into manageable segments, each delivering progress in increments. Collaboration with the third party was facilitated through emails and occasional zoom meetings.

Embracing Agile's adaptive planning and development also provided the flexibility to respond to changes, which is crucial in a cutting-edge field where new discoveries and knowledge alter the course of development. This methodology, with its emphasis on continuous learning and adjustment, was essential in navigating the challenges of this project.

## 4.3 Programming Languages

Python was chosen as the main programming language, as Python is a high-level programming language and is a popular choice for computer vision due to its extensive machine learning ecosystem, supportive community, and ability to handle large sensor data volumes. Its clear syntax facilitates rapid prototyping, while its flexibility aids in the iterative refinement of complex models. Additionally, Python's compatibility with various data formats and powerful visualization tools makes it ideal for processing and analysing the nuanced spatiotemporal aspects of 4D radar data, ensuring efficient development and insightful interpretation of object detection models.

## 4.4 Integrated Development Environment (IDEs)

The author chose PyCharm as is it is often the preferred IDE for implementing complex projects like implanting a PV-RCNN architecture for 4D radar due to its comprehensive Python-centric features. It offers an integrated environment tailored for Python development, including smart code navigation, advanced debugging, and refactoring tools, which significantly enhance productivity and code quality. PyCharm's support for scientific libraries and frameworks, such as TensorFlow and PyTorch, streamlines the machine learning workflow, from data exploration to model training and evaluation. The IDE's virtual environment management simplifies dependency handling, ensuring project consistency. Moreover, PyCharm's powerful visualization capabilities, coupled with its interactive Python console, facilitate the examination and interpretation of 4D radar data, making it an invaluable tool for researchers and developers working on advanced radar object detection systems.

## 4.5 Libraries

Providing detailed descriptions and explanations for each library would lead to an excessively lengthy manuscript, necessitating a more concise presentation.

Here is a brief overview of the core libraries:

Previously mentioned in **section 3.3 Open3d** was chosen for visualising the point cloud.

**Numpy**: A fundamental package for scientific computing with Python, providing support for large, multi-dimensional arrays and matrices, along with a collection of mathematical functions to operate on these arrays.

**LLVMLite**: A lightweight, dependency-free library for binding to LLVM's C++ API, enabling easy JIT compilation of numeric code.

**Numba**: An open-source JIT compiler that translates a subset of Python and NumPy code into fast machine code, significantly accelerating execution.

**PyTorch (Version 1.1):** A deep learning library that provides a flexible and powerful array library, Tensor, with GPU acceleration and automatic differentiation capabilities for building and training neural networks.

**TensorboardX**: An extension to TensorBoard, providing visualization and tooling needed for machine learning experimentation, such as tracking and visualizing metrics, and model graphs.

**PyYAML:** A YAML parser and emitter for Python, enabling easy reading and writing of YAML files for configuration or data serialization.

**Scikit-image**: A collection of algorithms for image processing in Python, providing tools for image manipulation and analysis.

**Tqdm**: A fast, extensible progress bar for loops and command-line programs in Python, enhancing the user interface and experience.

**Torchvision**: A package consisting of popular datasets, model architectures, and common image transformations for computer vision.

**SharedArray**: A Python module for sharing NumPy arrays between processes, useful for parallel and distributed computing.

**OpenCV-Python**: A Python wrapper for OpenCV, offering access to a wide range of image processing and computer vision functions.

**Pyquaternion**: A module for quaternion mathematics and 3D rotations, useful in tasks involving spatial transformations and 3D geometry.

## 4.6 Other Tools

**Anaconda Prompt and Virtual Environments**

Conda virtual environments are integral to Python development for managing dependencies in isolated settings, ensuring projects with differing requirements can coexist without conflicts. Conda's user-friendly interface simplifies complex dependency management, particularly beneficial in data science and machine learning domains where projects often depend on a blend of Python and non-Python libraries.

**Linux on Ubuntu Virtual Machine**

After an initial attempt was made to setup the OpenPCDet repository on a windows system the author found a resource that stated this repository is only supported on Linux, despite there being no mention of this in their repository and a lot of lost time the author needed to setup Linux on an Ubuntu Virtual Machine.

OpenPCDet, a library tailored for point cloud detection tasks in autonomous driving, operates on Linux due to its reliance on Linux-specific libraries, tools, and a development environment that's closely aligned with the needs of high-performance computing tasks, such as those involving CUDA for GPU acceleration.

## 4.7 Data Loader & Split

To facilitate model training and validation on the dataset which is 26.1 GB in total, it is partitioned into training and testing subsets. This partitioning is guided by predefined indices listed in 'train.txt' and 'test.txt' files, ensuring a consistent and reproducible split across different experimental runs. This approach adheres to best practices in machine learning by preventing data leakage and ensuring the model's performance is evaluated on unseen data.

A screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generated

**Fig 16:** Data Loader output

Illustrated by **Fig 16:** An output for the number of labels loaded for each .bin file in the training set was given. Then the loader gets a total number of the labels loaded for the training set, along with the number of points in the first point cloud. Lastly the loader attempts to load in the testing set and prints out the number of the points in the first .bin file.

A screenshot of a computer

Description automatically generated

**Fig 17:** Task Manager screenshot

Loading and visualizing the large-scale point cloud data consumes substantial memory, as evident from **Fig 17:** screenshot showing 60% usage of the available 16 GB RAM. This is due to the storage needs of the extensive data and its labels in RAM, necessary for real-time processing and visualization. The considerable memory footprint is influenced by the data's volume and complexity, along with the memory overhead of the utilized data structures.

## 4.8 Point Cloud Visualization in Open3d

A group of small colored squares

Description automatically generated with medium confidence

**Fig 14:** Visualization of radar point cloud for Frame 01047

label information is being interpreted incorrectly and used to create the bounding boxes in the 3D visualization.

A colorful objects flying in the air

Description automatically generated with medium confidence

**Fig 14:** Visualization of radar point cloud for Frame 01047

A person riding a bicycle on a street

Description automatically generated

**Fig 15:** Camera output for Frame 01047

**Some of the errors due to dependencies….**

A screenshot of a computer program

Description automatically generated

A screenshot of a computer program

Description automatically generated

A screenshot of a computer program

Description automatically generated

# Appendices

‌

Zhou, Y., Liu, L., Zhao, H., López-Benítez, M., Yu, L. and Yue, Y. (2022). Towards Deep Radar Perception for Autonomous Driving: Datasets, Methods, and Challenges. *Sensors*, [online] 22(11), pp.4208–4208. doi:https://doi.org/10.3390/s22114208.

2.0 A Vision for Safety AUTOMATED DRIVING SYSTEMS. (n.d.). Available at: https://www.nhtsa.gov/sites/nhtsa.gov/files/documents/13069a-ads2.0\_090617\_v9a\_tag.pdf.

Anwesh Marwade (2020). *Kalman Filtering: An Intuitive Guide Based on Bayesian Approach*. [online] Medium. Available at: https://towardsdatascience.com/kalman-filtering-an-intuitive-guide-based-on-bayesian-approach-49c78b843ac7 [Accessed 10 Oct. 2023].

AR, A. (2021). *LiDAR vs RADAR: Detection, Tracking, and Imaging - For 4D Sensing, AI, AR, AV... - Medium*. [online] Medium. Available at: https://4sense.medium.com/lidar-vs-radar-detection-tracking-and-imaging-ca528c0e9aae [Accessed 24 Oct. 2023].

AUTOCRYPT. (2023). *The State of Level 3 Autonomous Driving in 2023 | AUTOCRYPT*. [online] Available at: https://autocrypt.io/the-state-of-level-3-autonomous-driving-in-2023/ [Accessed 3 Oct. 2023].

Avijeet Biswal (2020). *Power of Recurrent Neural Networks (RNN): Revolutionizing AI*. [online] Simplilearn.com. Available at: https://www.simplilearn.com/tutorials/deep-learning-tutorial/rnn [Accessed 1 Nov. 2023].

baeldung (2022). *The Viola-Jones Algorithm | Baeldung on Computer Science*. [online] Baeldung on Computer Science. Available at: https://www.baeldung.com/cs/viola-jones-algorithm [Accessed 25 Oct. 2023].

Balasubramaniam, A. and Pasricha, S. (2022). *Object Detection in Autonomous Vehicles: Status and Open Challenges*. [online] ResearchGate. Available at: https://www.researchgate.net/publication/357953408\_Object\_Detection\_in\_Autonomous\_Vehicles\_Status\_and\_Open\_Challenges#pf2 [Accessed 25 Oct. 2023].

Balasubramaniam, A. and Pasricha, S. (n.d.). *Object Detection in Autonomous Vehicles: Status and Open Challenges*. [online] Available at: https://arxiv.org/ftp/arxiv/papers/2201/2201.07706.pdf#:~:text=Object%20detection%20consists%20of%20two

Barnard, M. (2016). *Tesla & Google Disagree About LIDAR - Which Is Right? - CleanTechnica*. [online] CleanTechnica. Available at: https://cleantechnica.com/2016/07/29/tesla-google-disagree-lidar-right/ [Accessed 10 Oct. 2023].

Bloom, C. (2020). *Introduction to Radar*. [online] Arrow.com. Available at: https://www.arrow.com/en/research-and-events/articles/introduction-to-radar [Accessed 23 Oct. 2023].

Borah, C. (2020). *Evolution of Object Detection - Analytics Vidhya - Medium*. [online] Medium. Available at: https://medium.com/analytics-vidhya/evolution-of-object-detection-582259d2aa9b [Accessed 25 Oct. 2023].

Cohen, J. (2020). *How RADARs work*. [online] Welcome to The Library! Available at: https://www.thinkautonomous.ai/blog/how-radars-work/ [Accessed 4 Oct. 2023].

Cohen, J. (2023). *4D LiDARs vs 4D RADARs — Why the LiDAR vs RADAR comparison is more relevant today than ever*. [online] Welcome to The Library! Available at: https://www.thinkautonomous.ai/blog/fmcw-lidars-vs-imaging-radars/#:~:text=4D%20RADARs%20work%20using%20MIMO,have%20a%20pretty%20bad%20resolution. [Accessed 10 Oct. 2023].

Davies, E.R. (2022). The dramatically changing face of computer vision. *Elsevier eBooks*, [online] pp.1–91. doi:https://doi.org/10.1016/b978-0-12-822109-9.00010-2.

Doppler effect | Definition, Example, & Facts | Britannica. (2023). In: *Encyclopædia Britannica*. [online] Available at: https://www.britannica.com/science/Doppler-effect [Accessed 4 Oct. 2023].

Everythingrf.com. (2021). *What are 4D Radars? - everything RF*. [online] Available at: https://www.everythingrf.com/community/what-are-4d-radars [Accessed 10 Oct. 2023].

Gandhi, R. (2018). *R-CNN, Fast R-CNN, Faster R-CNN, YOLO — Object Detection Algorithms*. [online] Medium. Available at: https://towardsdatascience.com/r-cnn-fast-r-cnn-faster-r-cnn-yolo-object-detection-algorithms-36d53571365e

GeeksforGeeks. (2018). *Introduction to Recurrent Neural Network*. [online] Available at: https://www.geeksforgeeks.org/introduction-to-recurrent-neural-network/ [Accessed 1 Nov. 2023].

Gibiansky, A. (2014). *Convolutional Neural Networks - Andrew Gibiansky*. [online] Gibiansky.com. Available at: https://andrew.gibiansky.com/blog/machine-learning/convolutional-neural-networks/#:~:text=If%20we%20use%20an%20m,)(j%2Bb).

Gillis, A.S., Burns, E. and Brush, K. (2023). *deep learning*. [online] Enterprise AI. Available at: https://www.techtarget.com/searchenterpriseai/definition/deep-learning-deep-neural-network

Girshick, R., Donahue, J., Darrell, T. and Malik, J. (2016). Region-Based Convolutional Networks for Accurate Object Detection and Segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 38(1), pp.142–158. doi:https://doi.org/10.1109/tpami.2015.2437384.

GitHub. (2023). *view-of-delft-dataset/figures/example\_frame\_2.png at main · tudelft-iv/view-of-delft-dataset*. [online] Available at: https://github.com/tudelft-iv/view-of-delft-dataset/blob/main/figures/example\_frame\_2.png [Accessed 24 Oct. 2023].

Hearst, M.A., Dumais, S.T., Osman, M., Platt, J. and Bernhard Schölkopf (1998). Support vector machines. *IEEE Intelligent Systems & Their Applications*, [online] 13(4), pp.18–28. doi:https://doi.org/10.1109/5254.708428.

Hesai Webmaster (2023). *What You Need to Know About Lidar: The Strengths and Limitations of Camera, Radar, and Lidar. | HESAI*. [online] HESAI. Available at: https://www.hesaitech.com/what-you-need-to-know-about-lidar-the-strengths-and-limitations-of-camera-radar-and-lidar/#:~:text=Compared%20to%20the%20traditional%203D,over%20100%2C000%20points%20per%20frame. [Accessed 24 Oct. 2023].

Jessica Van Brummelen, O’Brien, M., Gruyer, D. and Homayoun Najjaran (2018). Autonomous vehicle perception: The technology of today and tomorrow. *Transportation Research Part C-emerging Technologies*, [online] 89, pp.384–406. doi:https://doi.org/10.1016/j.trc.2018.02.012.

Levity.ai. (2023). *Deep Learning vs. Machine Learning – What’s The Difference?* [online] Available at: https://levity.ai/blog/difference-machine-learning-deep-learning

‌Mathworks.com. (2023). *What Is SLAM (Simultaneous Localization and Mapping) – MATLAB & Simulink*. [online] Available at: https://www.mathworks.com/discovery/slam.html#:~:text=SLAM%20(simultaneous%20localization%20and%20mapping)%20is%20a%20method%20used%20for,to%20map%20out%20unknown%20environments. [Accessed 10 Oct. 2023].

Navtech Radar. (2023). *FMCW Radar*. [online] Available at: https://navtechradar.com/explore/fmcw-radar/#:~:text=Frequency%20Modulated%20Continuous%20Wave%20(FMCW,by%20the%20radar%20and%20compared. [Accessed 10 Oct. 2023].

NHTSA. (2020). *NHTSA*. [online] Available at: https://www.nhtsa.gov/technology-innovation/automated-vehicles-safety [Accessed 23 Oct. 2023].

‌Ouaknine, A. (2022). *Deep learning for radar data exploitation of autonomous vehicle*. [online] arXiv.org. Available at: https://arxiv.org/abs/2203.08038

ResearchGate. (2019). *Figure 1. Schematic diagram of a basic convolutional neural network...* [online] Available at: https://www.researchgate.net/figure/Schematic-diagram-of-a-basic-convolutional-neural-network-CNN-architecture-26\_fig1\_336805909

Synopsys.com. (2023). *What is an Autonomous Car? – How Self-Driving Cars Work | Synopsys*. [online] Available at: https://www.synopsys.com/automotive/what-is-autonomous-car.html#:~:text=Definition,in%20the%20vehicle%20at%20all. [Accessed 10 Oct. 2023].

Tyagi, M. (2021). *HOG (Histogram of Oriented Gradients): An Overview - Towards Data Science*. [online] Medium. Available at: https://towardsdatascience.com/hog-histogram-of-oriented-gradients-67ecd887675f [Accessed 25 Oct. 2023].

Udemy. (2023). *Automotive Radar*. [online] Available at: https://www.udemy.com/course/automotive-radar-basics-to-advance/ [Accessed 10 Oct. 2023].

Xx, N., Xx, X. and Xxxx (n.d.). Millimeter Wave Sensing: A Review of Application Pipelines and Building Blocks. *IEEE SENSORS JOURNAL*, [online] (1). Available at: https://arxiv.org/pdf/2012.13664.pdf.

Yang, B., Guo, R., Liang, M., Casas, S. and Urtasun, R. (n.d.). *RadarNet: Exploiting Radar for Robust Perception of Dynamic Objects*. [online] Available at: https://arxiv.org/pdf/2007.14366.pdf.

Zhou, T., Yang, M., Jiang, K., Wong, H.T. and Yang, D. (2020). MMW Radar-Based Technologies in Autonomous Driving: A Review. *Sensors*, [online] 20(24), pp.7283–7283. doi:https://doi.org/10.3390/s20247283.

Zhou, Y. and Yue, Y. (2022). *Radar Signal Processing Fundamentals*. [online] Encyclopedia.pub. Available at: https://encyclopedia.pub/entry/23781 [Accessed 23 Oct. 2023].

‌Joel Markus Vaz and Balaji, S. (2021). Convolutional neural networks (CNNs): concepts and applications in pharmacogenomics. *Molecular Diversity*, [online] 25(3), pp.1569–1584. doi:https://doi.org/10.1007/s11030-021-10225-3.

‌ Deng, J., Dong, W., Socher, R., Li, L., Li, K. and Li, F. (2009). ImageNet: A large-scale hierarchical image database. *2009 IEEE Conference on Computer Vision and Pattern Recognition*. [online] doi:https://doi.org/10.1109/cvpr.2009.5206848.

‌ Hironobu Fujiyoshi, Hirakawa, T. and Yamashita, T. (2019). Deep learning-based image recognition for autonomous driving. *IATSS Research*, [online] 43(4), pp.244–252. doi:https://doi.org/10.1016/j.iatssr.2019.11.008.

‌Balasubramaniam, A. and Pasricha, S. (n.d.). *Object Detection in Autonomous Vehicles: Status and Open Challenges*. [online] Available at: https://arxiv.org/ftp/arxiv/papers/2201/2201.07706.pdf#:~:text=Object%20detection%20consists%20of%20two.

‌Nitin Kushwaha (2023). *A Brief History of the Evolution of Image Classification*. [online] Medium. Available at: https://python.plainenglish.io/a-brief-history-of-the-evolution-of-image-classification-402c63baf50

Deepchecks. (2021). *What is VGGNet | Deepchecks*. [online] Available at: https://deepchecks.com/glossary/vggnet/#:~:text=Its%20object%20recognition%20method%20developed,dataset%20by%20a%20wide%20margin.

‌Great Learning Team (2020). *Introduction to Resnet or Residual Network*. [online] Great Learning Blog: Free Resources what Matters to shape your Career! Available at: https://www.mygreatlearning.com/blog/resnet/#:~:text=ResNet%2C%20short%20for%20Residual%20Network,Residual%20Learning%20for%20Image%20Recognition%E2%80%9D.

‌Paperswithcode.com. (2020). *Papers with Code - DenseNet Explained*. [online] Available at: https://paperswithcode.com/method/densenet#:~:text=A%20DenseNet%20is%20a%20type,sizes)%20directly%20with%20each%20other.

‌Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J. and Houlsby, N. (2020). *An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale*. [online] arXiv.org. Available at: https://arxiv.org/abs/2010.11929 .

‌

GitHub. (2024). tudelft-iv/view-of-delft-dataset: This repository shares the documentation and development kit of the View of Delft automotive dataset. [online] Available at: <https://github.com/tudelft-iv/view-of-delft-dataset?tab=readme-ov-file#introduction>

kili-website. (2023). *Mean Average Precision (mAP): A Complete Guide*. [online] Available at: https://kili-technology.com/data-labeling/machine-learning/mean-average-precision-map-a-complete-guide

‌

‌ GitHub. (2024). *OpenPCDet/docs/multiple\_models\_demo.png at master · open-mmlab/OpenPCDet*. [online] Available at: https://github.com/open-mmlab/OpenPCDet/blob/master/docs/multiple\_models\_demo.png

‌

Guan, S., Wan, C. and Jiang, Y. (2023). *AV PV-RCNN: Improving 3D Object Detection with Adaptive Deformation and VectorPool Aggregation*. [online] ResearchGate. Available at: https://www.researchgate.net/publication/371321106\_AV\_PV-RCNN\_Improving\_3D\_Object\_Detection\_with\_Adaptive\_Deformation\_and\_VectorPool\_Aggregation

‌

‌

‌