Delft University of Technology
Master of Science Thesis in Embedded Systems

Counting People in a Group in Outdoor Scenarios using mm-wave Radar

Madhubanti Bagchi

TU Delft Delft University of Technology

Counting People in a Group in Outdoor Scenarios using mm-wave Radar

Master of Science Thesis in Embedded Systems

Networked Systems Group
Faculty of Electrical Engineering, Mathematics and Computer Science
Delft University of Technology
Van Mourik Broekmanweg 6, 2628 XE Delft, The Netherlands

Madhubanti Bagchi
m.bagchi@student.tudelft.nl
mbagchiofficial@gmail.com

10.11.2023

Author
Madhubanti Bagchi (m.bagchi@student.tudelft.nl)
(mbagchiofficial@gmail.com)
Title
Counting People in a Group in Outdoor Scenarios using mm-wave Radar
MSc Presentation Date
17.11.2023

Graduation Committee

Dr. Qun Song	Delft University of Technology
Dr. Marco Zúñiga	Delft University of Technology
Dr. Girish Vaidya	Delft University of Technology

Abstract

Recently non-image-based data capturing methods such as sensors like RF, ultrasonic or radars, Wi-Fi, Bluetooth, etc for People Counting (PC) applications have gained momentum as an alternative to camera-based systems due to the preference for privacy preservation. Among them mm-wave radars are a strong choice for data capture since they consume less power, they are robust to the extremes of weather and environment. Further, they record data in the form of a collection of points making them computationally lightweight. However, such type of data becomes a challenge when used outdoors since those environments are complex. It becomes extremely difficult to separately identify individuals from the collection of points. This makes PC outcomes inaccurate when using radar data.

Therefore in our project, we aimed to address these limitations of using mm-wave radar data. We treated groups of people as a unit, distinguishing them based on the number of people in the group. In this way, we were able to count the number of people by detecting the group size. To achieve this, we processed the raw radar output data made available by the AMS Institute into a dataset to suit our chosen deep learning model.

Through a literature survey, we found that computer vision algorithms based on Deep Neural Networks (DNN), detect objects while maintaining a balance between speed and accuracy. Moreover, DNNs are capable of extracting features more effectively than traditional Machine Learning models. We selected the YOLOv8 model by Ultralytics after concluding our literature survey, as the most suitable model for our research problem.

However, the selected model accepts three-channel images and videos to detect objects. Therefore, we customized the YOLOv8 model as well as formatted our radar data into a four-channel tensor input which would be acceptable by the model. We were successful in detecting groups of up to four people outdoors with a detection accuracy of 79.21%.

You can't change what happened. But you can still change what will happen. –
Sebastian Vettel

Preface

They say *It takes a village to raise a child*, I would say *It takes a village to fulfill a dream.* I am indebted to my supervisors Marco Zúñiga and Girish Vaidya for acting as my compass to navigate this journey of thesis. The people who financed my education, purely based on love and humanitarian motivation, have proved to me that humanity still exists. I am grateful to my family for supporting me in all my decisions. I would also mention my friends, partner, and cousins who literally ensured I had a roof above my head, food on my plate, and positivity around me while I worked during this journey. Lastly, I extend my gratitude to one man whose performance at the Grand Prix shows how to be the *smooth operator*, (read Carlos Sainz)!

Madhubanti Bagchi

Delft, The Netherlands
10th November 2023

Contents

Chapter 1

Introduction

Counting the number of people at a given location is among the foundational steps to formulate safety measures and evacuation plans for emergencies. For instance, during the COVID pandemic, people counting technologies helped many countries to restrict the formation of large crowds, subsequently prohibiting the spread of the infection[14][2]. Besides the surveillance aspect, analyzing the crowd densities in public areas helps to improve traffic management by suggesting alternative route plans. Moreover, PC provides data to analyze public preferences for visiting places in a region. Thus, the utility of people counting (PC) is diverse.

A good PC application would count the number of people as close to reality as well as in real-time irrespective of the density of the crowd. This demands precise identification of people from the scene and then counting them. Consequently, it is heavily dependent on the device used for data collection and also on the scene complexity.

The Closed Circuit Television (CCTV) is the most commonly used and ubiquitously present device for recording data for PC. There are one million CCTVs in the Netherlands making it rank among the top four countries in Europe with the highest amount of CCTVs, as per a report by the website Upcoming Security [50]. Generally, Computer Vision (CV) algorithms are used to detect people by their appearance and count them from the scene captured by these cameras. This results in encroaching on the privacy of an individual by recording his personally identifiable features while capturing his appearance[43].

Furthermore, if CCTVs are used utmost care needs to be taken to store and maintain the data and take necessary steps in case the data gets breached. This incurs additional costs and resources. Thus, the need for an alternative is strong where the privacy of an individual is preserved.

There are other technologies like infrared sensors, ultrasonic sensors, Bluetooth, Wi-Fi tracking, etc. But all of them have limitations either in their range of operation, or they record electronic device information of the people, or they are influenced by signal interference.

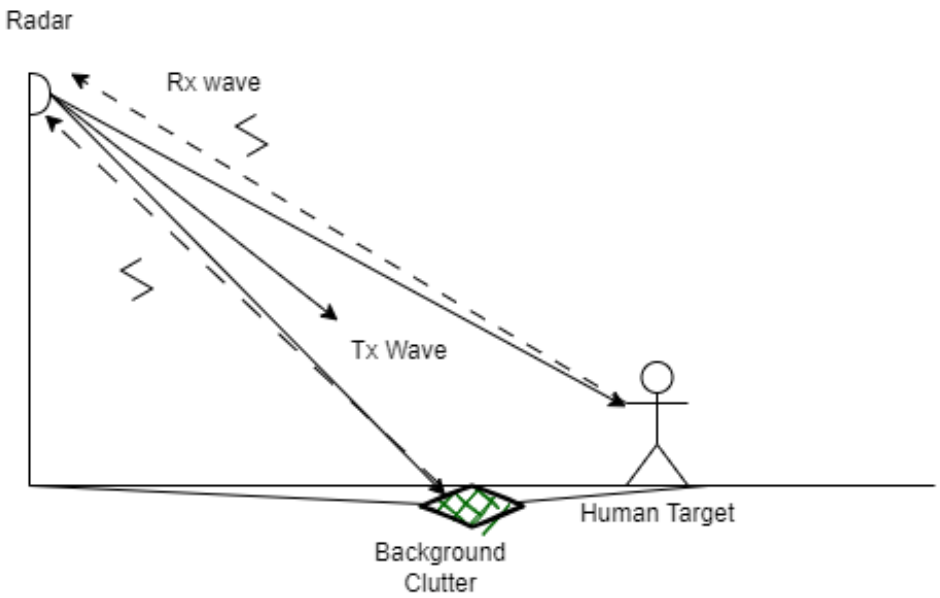


Figure 1.1: **Demonstration of the basic functionality of a radar. The radio waves are transmitted which in turn hits the target and gets reflected to the radar.**

Taking a step outside of existing PC technologies, the radars, especially the lidars and mm-wave radars have revolutionized the automotive industry with their target detection capabilities without recording personal information of any sort. For example, they are widely used in pedestrian and obstacle detection while driving.

A radar is a device that detects its target by transmitting radio waves at a certain frequency toward the direction of the target [51]. These waves are then reflected to the radar after hitting points on the surface of the target and are subsequently processed and stored as point cloud data. Hence, a target is represented as a cluster of points in the output of the radar. Each point in the cluster contains information about the target such as its range, angle, velocity, etc. This data is then interpreted using appropriate algorithms to detect the target.

The mm-wave radar, which operates in the millimeter wave bandwidth, is reliable in terms of its accuracy and precision in measuring the target's properties like velocity, range, and azimuth, irrespective of environmental conditions like lighting, dust, smog, harsh weather, etc. It is also cheaper compared to lidars and commercially available in free-license bandwidths for operations.

The number of points in a cluster for an mm-wave radar is also way lesser than the lidars, hence lowering the computational memory requirements. The mm-wave radars have been already used in various applications of human activity recognition [8] such as human fall detection[53], heart rate monitoring[28], as

well as people counting in indoor environments[24]. Thus, based on the multiple reasons discussed, in this project, we decided to use mm-wave radars to collect data for PC.

1.1 Challenges

Despite having multiple merits in favor of using mm-wave radars for PC, there are the following challenges as well.

- The outdoor environment is dynamic when compared to indoors, owing to a more complex scene with the addition of non-classified targets along with classified ones. This complexity limits the efficiency of algorithms that identify the target from radar output. Thus, there is a need to find an algorithm that would be robust to such dynamicity.
- It is easier to separately detect people from a group of closely spaced people when they are represented by an image than when represented by a cluster of points. To the radar, the group is one unified cluster of points. This poses a serious challenge in determining the count of people from radar data.
- The data generated by mm-wave is approximately 100 times less dense than that generated by Lidar owing to low resolution[34]. This low resolution of data results in coarse-grained applications, regardless of its accurate measurements of the target's properties.
- Moreover, several transmitted waves hit the same point on the target surface, resulting in many points having the same location coordinate. This contributes to a challenge when we try to plot the clusters because these points overlap and reduce the number of effective points in the cluster that can be used for detection purposes.
- The biggest pressing issue is the lack of publicly available datasets of mm-wave radars with data specific to our problem. This is a challenge at the basic level to start developing a PC application using the mm-wave radar.
- Usually Deep Learning (DL) models like Pointnet++, PointRCNN, and mmPointGNN have been used on the point clusters of mm-wave radar to detect human gestures and activity. But it is restricted mostly to indoors or used for an individual's activity recognition.

1.2 Key Problem Points to Solve

To simplify the problems discussed above

1. There are no publicly available datasets with point clusters presenting groups of humans outdoors. Hence, even to start to build a PC application, first, data are required.
2. Identify an algorithm that can work with sparse data and can effectively detect the target from the point cluster.

3. Devise a strategy to make the radar output suitable for the chosen algorithm as its input.

Two founding strategies were set based on which the thesis was conducted. Firstly, instead of counting individuals from the point cluster, each cluster has been treated as groups of humans with the group strength ranging from one to four. Anything greater was considered as five for simplicity.

Secondly, we selected a computer vision algorithm for classifying the clusters over other models. CV algorithms are designed keeping the speed of detection in mind since they are used in real to near-to-real-time applications.

Motivated by the challenges discussed in Section 1.2, we explored the existing literature to identify a deep learning model in the computer vision domain that can operate on sparse data and leverage feature extraction from the radar output while keeping a balance between speed and accuracy of detection.

We selected the YOLOv8 [16] by Ultralytics after concluding the literature survey. YOLOv8 is a state-of-the-art model for object detection. It works with even small datasets because it has image transforms implemented during data loading which increases the variety of instances and also makes it robust to noise. YOLOv8 has attributes that enhance feature extraction in multiple scales. This makes it effective for low-resolution data like that generated by the mm-wave radar.

We primarily focused on detecting the group strength of people closely spaced outdoors by using the data that the radar generated after encountering them. However, the consequent challenge that we faced by choosing a CV algorithm to detect the non-visual data, was to define an approach that could be used to convert radar data to a format accepted by the YOLOv8 model without compromising on its functionality.

1.3 Contributions

The following are the contributions of this research project to cater to the problems discussed above.

1. We created a dataset using the mm-wave radar output of closely spaced individuals walking in groups on the street that we received from the AMS Institute and classified them according to their group strength.

2. We developed a procedure to format the radar output into YOLOv8 acceptable format for images.
   Then we further extended it into a four-channel tensor and customized the preprocessing part of YOLOv8 model to process the same. We achieved an accuracy of 79.21% in detecting the strength of the group of humans using that input format for YOLOv8.

3. We formulated a strategy to retain the information of the multiple points tagged to the same location.

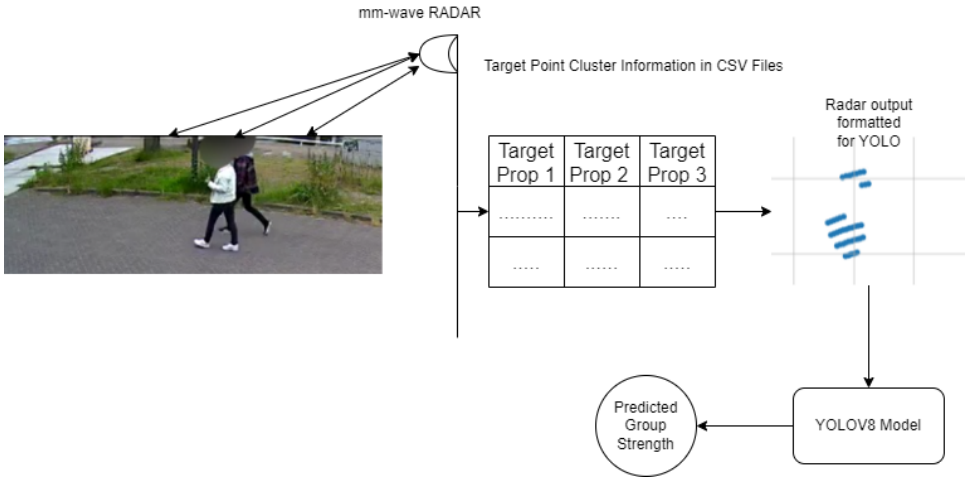


Figure 1.2: **Pipeline for the application. Starting from receiving CSV files containing point cloud data from the radar to detecting the number of people in a timestamp.**

Figure 1.2 shows the basic pipeline of the model. The radar collects data and then stores it in the form of CSV files. We utilize these radar data and transform them into a four-channel tensor which we then input to the YOLOv8 model. This model then detects the target and predicts the outcome as the group's strength. We discussed the same in detail in the following chapters.

In Chapter 2 we explored and analyzed the existing work on people counting and that of mm-wave radar. Following this, we identified our DL algorithm YOLOv8 which we use to detect how many people are there in a group.

Next in Chapter 3 we introduced the basic information related to radars and the YOLOv8 model as System Basics. Following this, in Chapter 4, we discussed the step-by-step details of the dataset preparation and creation.

Chapter 5 contains the discussion regarding the various aspects and approaches to model the radar features as an image and as well as a four-channel tensor to provide as input to the DL model.

In Chapter 6 we recorded the evaluation of the experiments following which we concluded in Chapter 7 along with future work proposals.

Chapter 2

Literature Review

In Chapter 2, we document the literary survey we conducted to explore and understand the existing work on camera and people detection methods, followed by the non-visual methods used for people detection. Finally, we review the various applications of mm-wave radars for human activity detection and identify the potential gaps in the existing applications that pose a challenge for using the device for PC.

Usually, computer vision models that are used for camera data are known for their speed and accuracy of detection. Hence in this project, we focused on selecting a computer vision model to investigate if we could successfully implement it on radar data. We highlight the obstacles we are met with in implementing radar data into a computer vision suitable input.

2.1 Camera and Computer Vision Algorithms

PC, which interchangeably is also called crowd counting, is a demand-driven research area owing to its diverse utility. PC using images and videos obtained through cameras covers a vast portion of research interest in computer vision (CV). People Counting falls under Multi-Object Tracking (MOT) in Computer Vision and Artificial Intelligence [29]. Traditional machine learning algorithms like Support Vector Machine (SVM), Random Forest, etc use hand-crafted features to detect people [23][49][27]. This limits the unlocking of all the possible potential features that would help in accurate detection.

The biggest drawback of these traditional machine learning algorithms is that their efficiency degrades fast when the scene becomes complex[19]. This can be due to multiple reasons like uneven light variation, high people density, changing scenes, or high occlusion. Hence, a more robust approach is considered with Deep Learning algorithms. DL, especially Deep Convolution Neural Networks (DCNN) are efficient in feature extraction and hence can deal with complex scenes more efficiently.

One important point to note is that during a conventional image classification task, the location of the object does not matter. This is understandable

from the point of view of a regular image. But in the case of radar data, the distance of the object from the radar influences the area of exposure and hence the measurement of the various properties of the target. Thus to add the position information, we considered object detection models.

Some common state-of-the-art algorithms for object detection are based on Faster R-CNN [37], and Mask-RCNN [11], You Only Look Once version 3 (`YOLOv3`) [36] and Single Shot Detection (`SSD`). The Faster-RCNN is based on the Fast-RCNN model but is more improved in terms of accuracy and speed. The Faster-RCNN proposes a neural network called a Region Proposal Network (RPN) which is dedicated to proposing potential object locations in the entire image. This speeds up the object detection which is done using Recurrent Convolutional Neural Networks (RCNN). But the downside of this model is that it is computationally demanding thus negating its choice for real-time applications.

Faster RCNN works in two stages. But, later single-stage detectors (SSD) have been innovated which does detection in a single forward pass through the network. This in return has made real-time object detection more efficient. The architecture of SSD is simpler compared to Faster-RCNN making it computationally more faster and lighter. But Faster-RCNN has better accuracy than the SSD model. Hence we could see that the balance of speed and accuracy is always a challenge in these object detection models.

The latest developments in the CV domain have been made by the object detection model called You Look Only Once (YOLO). This is also a one-stage object detector that locates the object and also provides the class probabilities in one forward pass of the network. The YOLO model fuses the concept of Region Proposal and object detection into a single step making it much faster than the existing Faster-RCNN model and more accurate than the conventional SSD detector. The YOLO model has several versions and the most recent one is the YOLO version 8 by the Ultralytics Company. The code is open-source and freely accessible in the public domain. One can use it and modify it according to their specific requirements by citing due credits. This prompted us to consider YOLOv8 as a potential model among the CV algorithms.

There are already DL models for PC that have been developed based on Neural Networks (NN) e.g., CrowdCNN [54] and CompactCNN[42]. CrowdCNN is one of the earliest models of PC using CNN and is not efficient in handling densely populated scenes. On the other hand, CompactCNN is devised to handle dense populations and it works keeping the balance between speed and accuracy. Transfer learning models have also proved to be beneficial in such crowd-counting applications as is the case of CSRNet[25]. But these works are structured and dedicated to camera data. The algorithms are also not open-source which makes it difficult to improvise for radar and investigate their potentiality.

Thus, we concluded that to create a PC algorithm that works on the radar data we could consider the state-of-the-art object detection models and customize it to suit our requirements.

2.2 Non-visual People Counting

Though major research of `MOT` is based on image and video processing, currently non-imaged-based people counting has also gained momentum in the field of digital surveillance, especially in human detection and counting. RFID, Wi-Fi, radars, etc are a few technologies that do not capture the actual image of the people yet can count their presence[20]. One major disadvantage of RFID or CSI (`Channel State Information`)[57]is that they detect the presence of an object through the device present with the target object. This again makes the application dependent on the target and compromises the device privacy of the target. On the contrary, Wi-Fi Sensing or radar technology works based on information received through signal reflection. The limitation of Wi-Fi Sensing lies in its range of operation and resolution [18].

2.3 Radars and Lidars

This limitation of the discussed devices used for collecting data can be overcome with the use of radars. The main advantage of radars is their non-intrusive object detection capability. It does not identify or use the actual image of the individual and hence maintains the person's privacy. The target doesn't have to wear any device to mark its presence unlike in RFID. The radar provides accurate velocity and distance estimation of the target, unlike other non-image-based tracking options. Also, the object detection range in RADAR is higher than that of RFID or Wi-Fi.

The most popular radar used for object detection is lidar[58]. Lidars produce rich densely populated point cloud data which achieves good accuracy in identifying the target object. They have been used extensively in the automotive industry for obstacle detection. The merit of lidar's object detection has made its place in PC technologies as well [10]. Though lidar is very accurate in detecting the object it has some downsides. Mainly the lidar is not very accurate in harsh weather conditions or dusty environments. Secondly, it is a very expensive device. The lidar cannot operate under poorly lit conditions.
The mm-wave RADAR is another type of radar that operates using millimeter waves for object detection. Hence it is robust to anomalous weather and environmental situations. Also, it is cheaper in cost in comparison to the lidar.

2.4 mm-wave Radars and applications

The mmWave Radar is primarily utilized in the automotive industry for autonomous driving and in-vehicle surveillance. [4][52][1][44]. Owing to its low cost, low power consumption and high frequency of operation, the mmWave radars are growing in popularity for exploring more accurate methods of human detection and counting[6]. The output from the mm-wave radar is the collection of point cloud data which automatically reduces the memory size leading to the opportunity of more processing capacity. Various methods have been implemented to exploit the point cloud data as discussed in [39] [38].

2.5 mm-wave and Deep Learning

The mm-wave radars have been used in Human Activity Recognition, Gesture Recognition, human detection, and tracking [9][55][12]. When looking into the algorithmic models implemented to utilize the radar data and achieve the application goals, the methods range from conventional algorithms to general machine learning algorithms. However, the challenge is the scarce amount of data obtained using mm-wave does not give high accuracy with such models. Deep Learning and specifically Neural Networks are being explored to extract more features from the point cloud data.

[41], proposes an approach for PC inside a vehicle by using Deep Metric Learning. The paper introduces a novel loss function called the Label-Aware Ranked loss which is used to calculate the rank or order of the labels of the data points of the radar point data.

In [9], the mmPointGNN does human activity recognition by creating a graph from the points as nodes and performing convolution in the edges on the target state properties. The edges are dynamic, meaning they are not always connected to the same data points. This unique feature enables the prediction of varying gestures and movement patterns. The merit of this model has been verified indoors and with single human activity. This model fails to recognize multiple humans when closely spaced. This is because it is difficult to identify and then separate nodes in the context of single individuals.

From the perspective of point data, the PointNet [7] and PointNet++ [35] models are the two basic classification models. The PointNet++ is an extension of the PointNet model. Unlike PointNet, PointNet++ has better feature extraction methods and incorporates local information of the image during processing.

On the other hand [33], uses a combination of Pointnet++ blocks with LSTM to utilize the sparse mm-wave point cloud to recognize mid-air hand gestures. Another important example is [56], where the authors use bi-directional LSTM in combination with Deep Recurrent Neural Network to identify humans and then use a Random Forest Classifier to classify the human as one of the known individuals. But all these applications degrade in application efficiency when in the outdoors as well as when multiple people are closely spaced.

In our project, we received the radar data of closely spaced humans outdoors from the AMS Institute. Hence, we explored the existing work to select an approach that would work efficiently with the given type of data. In this project, we decided to try CV algorithms due to a couple of motivation factors.

Firstly, CV algorithms bring a balance in the speed and accuracy of object detection. As discussed previously we selected the object detection model over the classification ones because we wanted to incorporate the positional information of the target.

Secondly, object detection models have efficient feature extraction blocks to

extract features even in low-resolution input. This acts in our favour since mm-wave radar data is significantly sparse.

Another factor that motivated us to look into the CV algorithms was its robustness to a low amount of training data. Current object detection models are equipped to handle class imbalance and low amounts of training data. They use transforms to augment training data by increasing variety and thus improving the training process.

There are a variety of CV algorithms and approaches that are available for exploration. Among them, we selected the YOLOv8 by Ultralytics. This is because the YOLOv8 is currently being used in several applications and hence can be considered as a potential starting point. Secondly, the model has pre-processing methods to augment training data and increase its variety. Thirdly, the code is accessible in the public domain and hence can be customized to suit the requirements of the application problem. In addition to this, its previous version YOLOv5 has been used in mmwave applications in the automotive industry[21]. Though that application uses heatmap and hence utilizes only two properties of the radar data, this chain of information helped us to select YOLOv8 as the starting point to detect the count of people in a group from mm-wave radar output data.

Chapter 3

System Basics and Background

Before getting into the details of the work done in the project, we discuss the definitions and overview of the basics of radar and YOLOv8 model in this chapter. We define the properties measured by the radar that help to identify the target state as its state properties. The state properties of a target are its Range, Azimuth, Doppler, and Signal-to-Noise-Ratio (SNR). These properties measured by the radar, help to distinguish the target. Then we provide an orientation towards the architecture of the YOLOv8 model and how it operates.

3.1 Radar Basics and IWR6843ISK

In this section, the basic terminologies are discussed that are intrinsic to the radar measurements.
The radar receives waves that hit the target surface and get reflected from it and thus collect information about the target through these waves. The state of the target is defined by its properties like range, azimuth, Doppler velocity, and SNR which are carried back to the radar by these received signals. These signals are represented as points and each point holds the state information.

A point cloud is a collection of data points that hold spatial as well as other information such as Range, Azimuth, SNR, and Doppler of the target. Throughout the report, the term point cloud and point cluster have been used interchangeably. Figure 3.1 shows a visual demonstration of the point cloud.

3.1.1 Target State Properties

Signal-To-Noise-Ratio (SNR)

This property provides an estimate of the quality of the received wave. The higher the SNR value, the stronger would be the signal strength. As a consequence, the points that have high SNR values can be considered to be more reliable and robust to the noise in terms of the information transmitted with the wave. This is an important property in deciding the authenticity of the

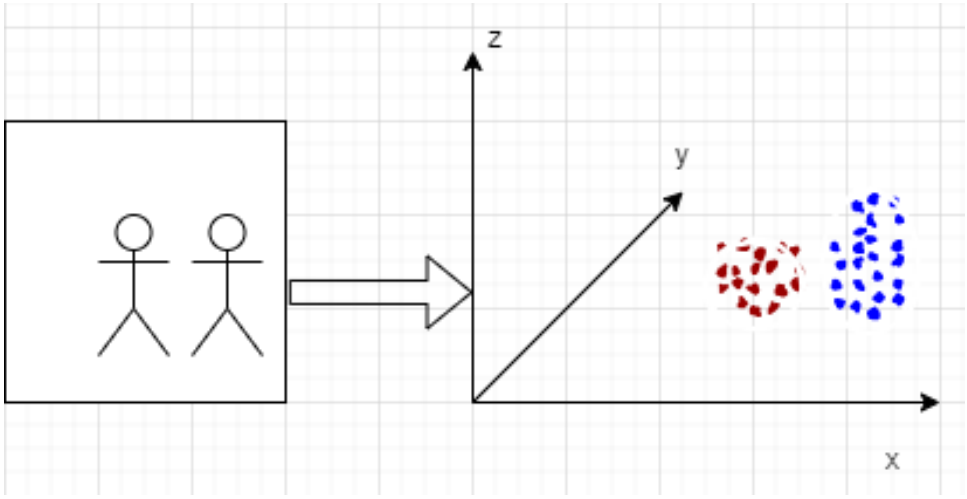


Figure 3.1: **The right image shows the point cloud representation of the people in the left image. The arrows indicate the coordinate axes.**

information.

The radar unit itself uses algorithms like Constant False Alarm Rate(CFAR) to filter out received signal information that is below a certain SNR threshold before generating the output. The SNR of a received wave can vary due to many reasons - if the target is very distant from the radar, the target surface is reflective or not, etc.

Range

This property is also directly obtained from the output of the radar. It is a measure of the distance of the object from the radar. It is important to consider the range while detecting a target because depending on this value, the exposure to the transmitted waves also differs. The other factors being the same, a closer target offers more radar cross section and thus a larger cluster size.

Azimuth

The azimuth value provides the angular resolution to differentiate targets with the same range value. The range and azimuth values are used to derive the x and y coordinates of the points in the cluster.

Doppler

One of the most important properties of the target is the Doppler frequency. It is from the Doppler frequency, the velocity of the target is estimated to the radar. In addition to the position (defined by range and azimuth) and velocity, the velocity of an object helps in its identification.

The property value that we receive from the radar with the name Doppler

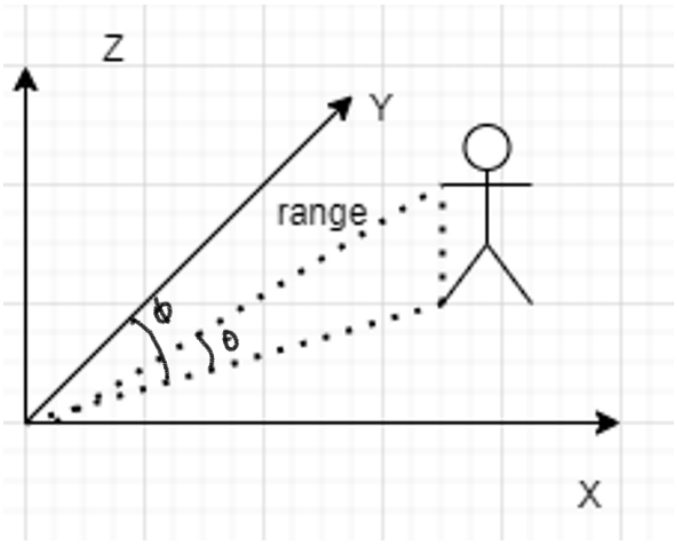


Figure 3.2: **The reference frame of the sensor is used always. The coordinate system is presented both in Cartesian (XYZ) and Polar coordinate systems where d is the Range, ϕ is the Azimuth and θ is the Elevation. In our project, the z-axis has been ignored because the image is considered a 2-D presentation.**

is the velocity with the unit of meters/second. Hence in this report from now onward, Doppler will refer to the relative velocity of the target to the radar.

Relation between Azimuth, Elevation, and Range with the Cartesian Coordinates

We were required to derive the x and y coordinates to convert point clouds into an image format. The x and y coordinates of the points in the point cloud were derived from the azimuth and range values from the radar using the following equations.

$$y = range * cos(azimuth) \tag{3.1}$$

$$x = range * sin(azimuth) \tag{3.2}$$

Figure 3.2 represents the relation of the azimuth and range with that of the x-y values. Though the radar output is produced from a 3-dimensional coordinate system, but we consider only the x and y coordinates since we format the data in a 2-dimensional image.

In addition to the above-discussed fundamental properties, one can derive statistical values from these properties. The below properties were also been considered.

Standard Deviation of SNR

This property is calculated based on the SNR values of the points in the cluster. This can be considered an important property since it shows how much the SNR

varies within the cluster.

Mean of Range

The mean of the range is calculated for the entire point cluster. The target in this project is humans and they are not of trivial size. Hence the mean range of all the points in the cluster can provide information regarding the target size.

Mean of Azimuth

Similar to the mean range, the mean azimuth provides a better representation of the azimuth than its values owing to the size of the target.

Standard Deviation of Doppler

The standard deviation of the velocity derived from the Doppler frequency represents the value for the entire cluster. Higher variability can indicate more number of people in the group.

Point Density

We derived this property by calculating the number of points having the same coordinate values.

3.1.2 IWR6843ISK

The mm-wave radar used to procure data for this thesis project was the IWR6843ISK from Texas Instruments. This radar board operates at a frequency of 60 GHz that lies in the license-free bandwidth. It has 4 receiver antennas and 3 transmitter antennas [13].

In terms of the field of view (FOV), it has an azimuth of 120° and 30° elevation. Figure 3.3 shows the model of the radar board used to collect data for this project. The IWR6843 has an existing algorithm called the GTrack algorithm which is further extended to count the number of people by calculating the number of active clusters that are being tracked. The model works mainly indoors for occupancy measurement.

3.2 YOLO - You Look Only Once

You Look Only Once is a computer vision model that was first introduced in 2015. It has been improved over the years and in 2023, the Ultralytics company proposed the version of this model i.e. YOLOv8[16]. In this thesis project, versions 8.0188 and 8.0163 were used to perform the experiments.

Though officially, this model version has not been published, it has already been used in several real-time applications successfully. The YOLOv8 is an anchor-free model built on the Pytorch framework[45]. Like any other computer vision model, it is divided into the backbone and head sections.

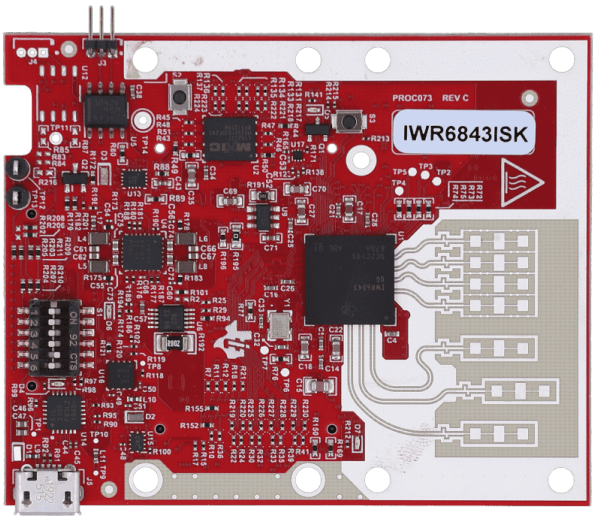


Figure 3.3: **IWR6843ISK board. [13]**

The backbone is responsible for extracting the features while the head takes care of the detection of the target. The primary merit of this model family is to strike the balance between achieving high-speed object detection and accuracy of detection.

The model sizes available for YOLOv8 are shown in figure 3.4. They all differ in the amount of computational memory requirement and the number of layers utilized in the architecture. In this project, the small and large versions were used to conduct the experiments.

The figure 3.5 is a simplified representation of the framework of version 8. The architecture is divided into neck and head sections.

3.2.1 Backbone

The backbone section is where the features are extracted from an image. YOLOv8 uses the Feature Pyramid Network [26] to extract features in layers from the global to the local scale. The backbone framework is a customized version of the CSPDarknet53[5] and it starts the feature extraction process with a convolution layer having a large window. This is aimed at reducing the memory and computational requirements.

Then the rest of the layers follow to extract features from varied scales. The C2f module is a cross-stage partial bottleneck with two convolutions. This module combines the high-level features with contextual information and thus improves the detection accuracy. The extracted features are then pooled in the SPPF layer (Spatial Pyramid Pooling Fast) which allows the model to gather features from various levels of abstraction [47].

Detection (COCO) | Detection (Open Images V7) | Segmentation (COCO) | Classification (ImageNet) >

Model	size (pixels)	mAP^{val} 50-95	Speed CPU ONNX (ms)	Speed A100 TensorRT (ms)	params (M)	FLOPs (B)
YOLOv8n	640	37.3	80.4	0.99	3.2	8.7
YOLOv8s	640	44.9	128.4	1.20	11.2	28.6
YOLOv8m	640	50.2	234.7	1.83	25.9	78.9
YOLOv8l	640	52.9	375.2	2.39	43.7	165.2
YOLOv8x	640	53.9	479.1	3.53	68.2	257.8

Figure 3.4: **Model sizes of YOLOv8 by Ultralytics. [13]**

3.2.2 Head

The head section of the model is where the detection, objectness, class, and box losses are calculated. The box loss also denoted as `box_loss` is the difference between the predicted bounding box and the ground truth bounding box. The class loss which is `cls_loss` is a measure of how accurately the classification of the predicted box was made. It is also referred to as the cross entropy loss[17].

YOLOv8 uses an anchor-free model with the head formulated in a way to process object detection tasks independently. This design facilitates the improvement of the model's overall accuracy since each task can be performed independently. The sigmoid function has been used in the output layer for the objectness score indicating the probability of the object within the bounding box.

It uses the softmax function for the class probabilities, representing the objects' probabilities belonging to each possible class. YOLOv8 uses Complete Intersection over Union (CIoU) instead of Mean Square Error (MSE) and Deformable convolution layer loss (DFL) loss functions for bounding box loss. The smooth L1 function has been used to calculate the regression loss [15]. These losses have improved object detection performance, particularly when dealing with smaller objects.

3.2.3 Bounding Box Calculation

It is necessary to annotate the instances used for training and validation purposes in the YOLO format. Bounding box is also termed as bbox and the same has been used in the thesis report. It is the box drawn around the target for its localization.

The YOLO format to represent a bounding box is as follows - Class index, x coordinate of the bbox center, y coordinate of the bbox center, the width of the bbox, the height of the bbox. Apart from the class index, all other values

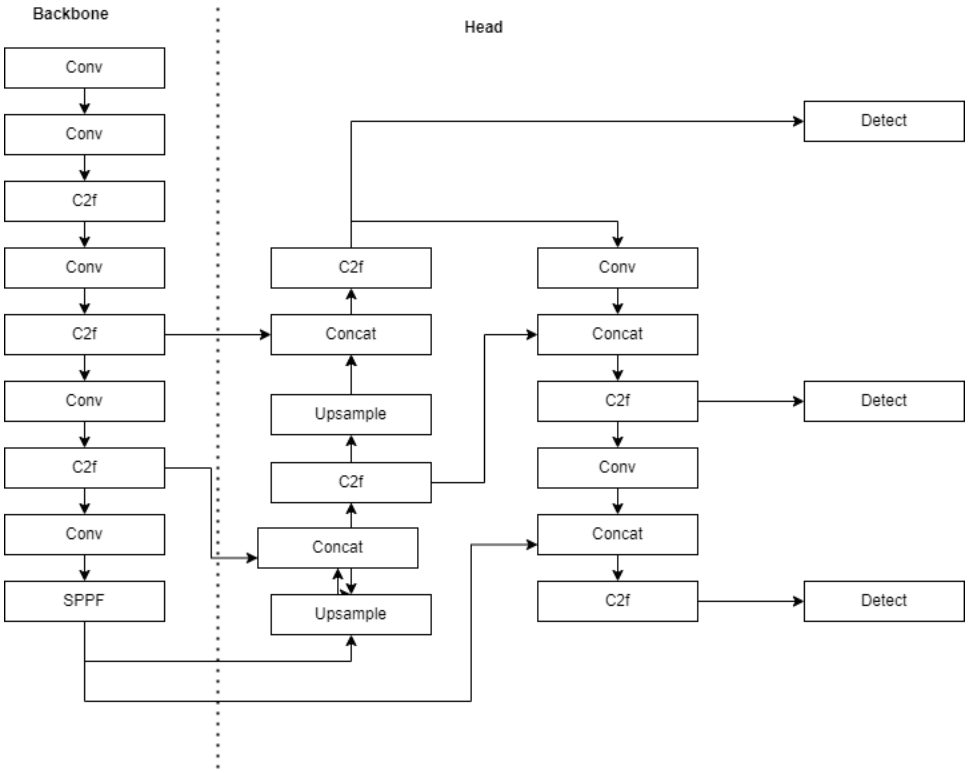


Figure 3.5: **Simplified Architecture Diagram of YOLOv8 by Ultralytics. The detailed architecture diagram is present in Appendix Section**

are normalized to the height and width of the image.

In this thesis, we have the x and y values of the points present in the point cluster. hence we calculate the bbox_x_center = x_min + (range of x)/2. Similarly, we calculate the y center of the bbox. The width and height of the bbox are the ranges of x and y respectively. Figure 3.6 shows the bounding box around the target in an image.

3.2.4 Input Transforms

The YOLO enhances the training efficiency by applying various image transforms to the training images while loading them. YOLOv8 imports the Albumentations class which contains image transforms to be used for data augmentation. A few of them are as follows and they all belong to the Albumentations class [3].

- Blur(p=0.01, blur_limit=(3, 7))
- MedianBlur(p=0.01, blur_limit=(3, 7))
- ToGray(p=0.01)

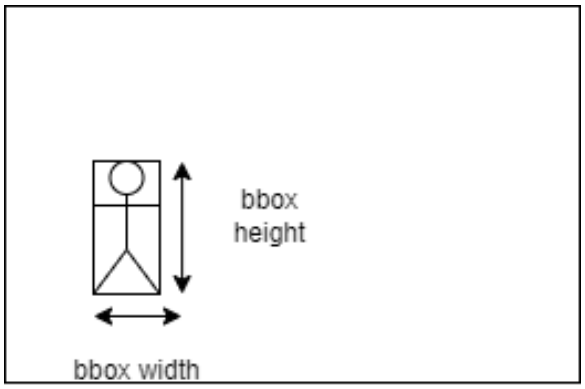


Figure 3.6: **Bbox around a target in a generic image. In this thesis, the bounding box values are calculated based on the x and y coordinate values of the point cluster.**

- CLAHE(p=0.01, clip_limit=(1, 4.0) - Contrast Limited Adaptive Histogram Equalization
- tile_grid_size=(8, 8))

These transforms make the training of the model robust to variations in the image making it suitable for object detection where the scale, background, and environment of the target can vary.

As we later extended the number of channels from three to four, the transforms ToGray and CLAHE had to be removed as they operate on one channel or three channels.

Chapter 4

Dataset Generation

Data is an important element in solving any Machine Learning problem. Hence, in this chapter, we discuss how we obtained the data for our research problem and explored the same to understand its characteristics to process it for our application. Usually, in most DL applications, the majority of the time is dedicated to understanding and exploring the data. This chapter documents the Data Collection and the Data Exploration blocks of our DL application pipeline as represented by Figure 6.1.

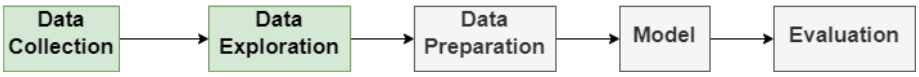


Figure 4.1: **The highlighted blocks, Data Collection, and Data Exploration are discussed in this chapter.**

4.1 Data Collection

The data collection was not a part of this project and has been done by the team at the AMS Institute. The Amsterdam Institute for Advanced Metropolitan Solutions (AMS Institute) is a research institute and was formed through a collaboration between academic institutions (including TU Delft) and Gemeente Amsterdam responsible for investigating solutions for deployment in urban infrastructure.

An mm-wave radar IWR6843ISK, was mounted facing a street behind the AMS Institute. In addition to that, a camera was also set up, both along the same axis, so that both collect data from the same perspective. Volunteers walked on the street as singles and in groups of two, three, four, and more people so that the radar reflections of humans outdoors could be captured. The radar point cloud data was stored in CSV files. The table 4.1 shows the parameters of the radar used to gather data.

Parameter	Value
Maximum range (m)	34.306
Maximum velocity (m/s)	7.969
Chirp Time (us)	36.411
Range resolution (m)	0.343
Velocity resolution (m/s)	0.125
Carrier Frequency (GHz)	60.290

Table 4.1: **The IWR6843ISK radar settings used by AMS Institute to procure the data for training, validation, and testing.**

4.2 Key Data Considerations

After receiving the data from the AMS Institute, we investigated the data to understand its nature. While processing the radar data, the following points were taken into account throughout the project.

1. The breadth of a street in general in old Amsterdam is 12 meters approximately. Hence the depth of the radar frame was considered to be 12 meters and the width as 15 meters as shown in figure 4.2.

2. All points with Doppler value as zero were excluded. Hence this thesis project focuses mainly on moving humans as its target of detection.

3. All points with negative SNR values were discarded.

4. For the simplicity of modelling, any group having more than four people was categorized as a group of five.

5. Bikes were considered as a distinct class in this thesis. Otherwise, the clusters of bigger groups could be confused with bikes as both of the classes have a large surface of exposure. Hence, an individual walking with a bike could generate a cluster equivalent to a larger group.

4.3 Categorizing Data

The data provided by the AMS Institute were organized based on the experiment day and type. We further categorized them into separate frames and instances by matching their timestamp with the equivalent timestamps of the camera data since that was our truth reference. The following classes have been created to categorize the data.

1. Class 0 : One Person (1)

2. Class 1 : Two Persons (2)

3. Class 2 : Three People (3)

4. Class 3 : Four People (4)

5. Class 4 : More than four people (5)

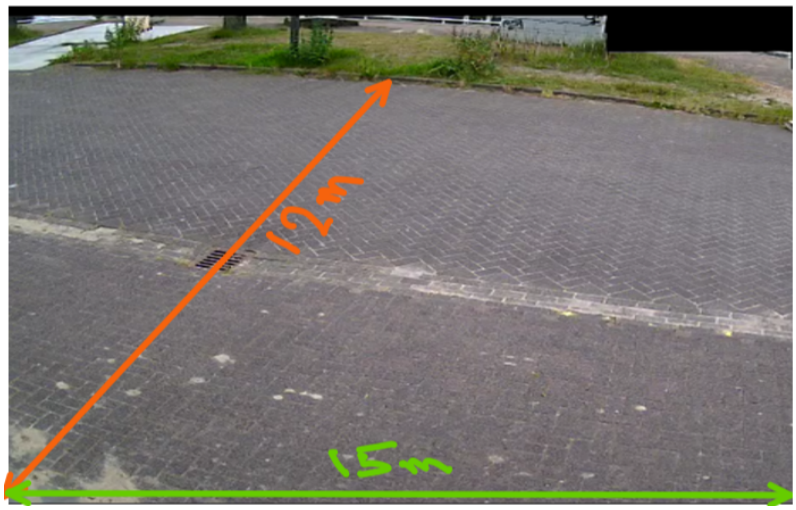


Figure 4.2: **A street near the AMS Institute in Amsterdam. The breadth of the street is 12 meters and thus the radar frame is 12 meters in depth and 15 meters in width representing a 2D visual representation of the street.**

6. Class 5 : Bikes

After establishing the classes, the next step was to extract the radar frames and subsequently the point clusters as instances from the radar output provided by the AMS Institute. To do the same, first, we plotted the frames from the timestamp CSV file as a scatter plot using Python. Then we manually selected the point clusters from the image using the LassoSelector Tool [30] offered by the matplotlib library of Python. This tool saves the points which lie inside the area drawn on the plot by the user. Figure 4.3 shows an example of plotting the point clusters and then matching the clusters with the human groups from the camera data.

It must be noted that there was a delay of about 10-15 seconds with the synchronization of the camera and radar data. This was because the clocks of the radar unit and the Raspberry Pi that was used to save the data in the cloud server were not the same and hence the timestamp recorded for the CSV files is that of the Raspberry Pi. To match the radar data to that of the camera, we first generated the plot and then started the video so that we could understand the starting timestamp concerning that of the camera. We added the time difference to every timestamp of the radar data to match it with its equivalent camera data.

Additionally, we saved the unclassified clusters separately which was used for the background images. Figure 4.4 shows the structure in which the data is stored.

Figure 4.3: **The radar output was marked into their respective classes using the Lassoselector Tool based on the camera data.**

4.4 Outlier Removal

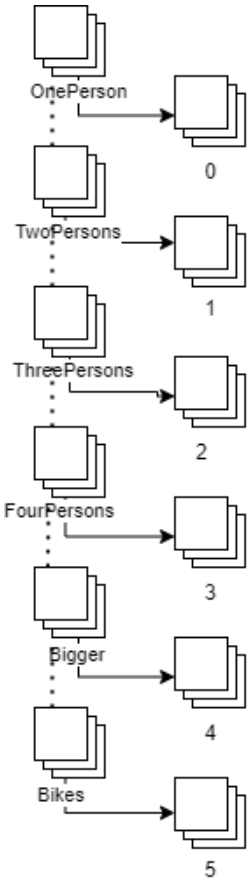
As the marking of the clusters to their respective classes was done manually, there was a possibility to include noisy points in the cluster data. Thus we refined the saved cluster information using the DBSCAN algorithm [40] especially because this algorithm is robust to noise. The cluster label "-1" indicates noise in DBSCAN.

There are two important parameters of the DBSCAN algorithm and they are the epsilon and the minimum sample values. The epsilon value is the maximum distance two neighbouring points can have. The minimum sample represents the minimum number of points required to form a cluster. We used the same epsilon and minimum sample values for all the classes. This is because different classes had different structures of the clusters and also within the same class, their shape varied greatly. This made it difficult to identify different epsilon values for different classes. The trade-off between having different epsilon values and having one was that additional outliers might still be present.

After several trials and errors, we finalized the epsilon value to be 1.5. For the minimum sample points, we found after calculating the number of points per instance that 19 is the lowest number of points in a cluster. Hence we finalized min_sample as 19.

4.5 Data Exploration

After sorting the data according to the classes, we then investigated the data based on the following aspects.



Instances Per Class	
Class 0	4489
Class 1	4958
Class 2	4249
Class 3	3516
Class 4	3931
Class 5	1173

Figure 4.4: **The CSV files folder of each class were stored separately in folders. When the input data for the model are generated, they are stored in the folders of the respective classes.**

4.5.1 Number of Points in the Cluster

Each point in the cluster is a reflected wave carrying information about the target after hitting a point on its surface. We explored the point cloud density of the data to assess the scarcity and also to determine the minimum sample number for the DBSCAN algorithm. Table 4.2 shows the point cluster strength for each of the classes individually.

4.5.2 Properties From the Raw Data

The CSV files contain a lot of information including the timestamp, the frame number, the cluster number derived from GTrack Algorithm, etc. Out of them, the most important properties are the SNR, Azimuth, Range, and Doppler. The frame number is also a vital property when it could be used for adding temporal context.

Class	Max NumPoints	Min NumPoints
0	644	20
1	1012	19
2	636	19
3	789	20
4	843	19
5	1012	20

Table 4.2: **The table shows the minimum and maximum number of points a cluster can consist of in all the classes.**

4.5.3 Relation of Range and Area of Cluster

Figure 4.5 shows the plots of the range (`x-axis`) with the area (`y-axis`) of the point cluster for each of the classes. We observed that the area value reached a peak for all the classes at some range value and then consistently started decreasing.

This was a vital observation that indicated that no matter how big or small the group of people was, the area depended on its distance from the radar. The area can also be considered as the surface area where the mm-waves hit and reflect to the radar. Thus more the target was well exposed to the radar, the greater its area.

Moreover, the area values before the peak were low despite being nearer to the radar. This was primarily because the mm-wave radar was mounted at an elevation of 3 meters and if the target was very close to the base of the radar, its area of exposure also reduced.

We inferred from this observation that an object detection model would be appropriate for our project because the position of the target influenced its area of exposure to the radar and thus the shape and size of the point cluster.

4.6 Data Augmentation using Transforms

It was evident from the instance count that class 5 could potentially create a class imbalance with only 1173 instances. Therefore this could generate bias during the model training. To overcome the class imbalance, we increased the instances of Class 5 as well as the overall training dataset using the following transforms.

- **Flip** - This transform flips the image or tensor as its mirror opposite. Both horizontal and vertical flips were used in this project. We used this both for image and tensor inputs.
- **Random Erasing** - As the name suggests, the transform function removes a portion of the object. The entire bounding box remains as the original. Such kinds of transforms are used to handle occlusion in the scenes.

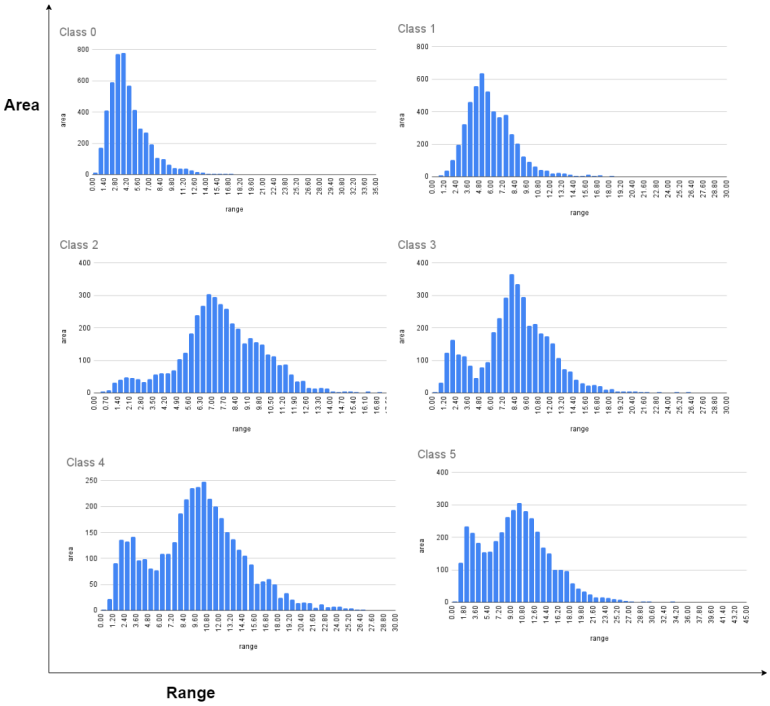


Figure 4.5: **The area of the point clusters is plotted against the range for each of the classes. The order is from the top left to the bottom right as Class 0, 1, 2, 3, 4, 5.**

- **Gaussian Noise** - We added random Gaussian Noise to the images and tensors to make the input similar to the dynamic real-life environment.

- **Histogram Equalization** - HE is used to enhance the contrast of pixel intensities in images to improve the visibility of details. This is achieved by calculating the histogram of the frequency of the various pixel intensities followed by normalizing the Cumulative Distributive Function of the same.

4.7 Data Division Information

The number of instances in each class had some differences. Therefore, we first fixed the number of instances for testing from each class and then divided the dataset into training and validation. This was primarily motivated by the fact that the accuracy of the testing required to be an unbiased one. Hence for testing, 300 instances from each class were selected, and then the testing accuracy was calculated. Then we split the remaining dataset into 80% and 20% for training and validation datasets respectively. The table 4.3 shows the total data and the division of the instances in the training, validation, and test sets.

Class	0	1	2	3	4	5
Training	3142	3470	2974	2461	2751	2246
Validation	897	991	849	703	786	673
Test	300	300	300	300	300	300

Table 4.3: **The division of the instances per class has been shown for training, validation, and testing.**

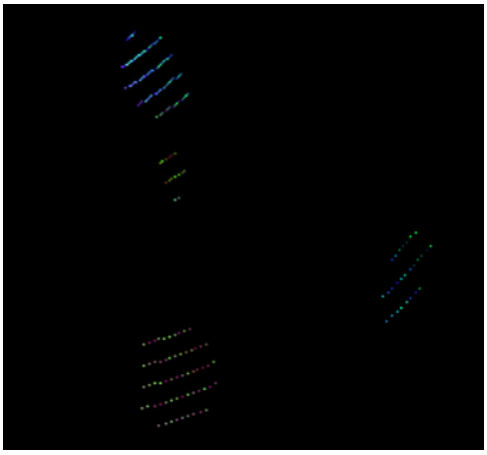


Figure 4.6: **This is an example of a background image. This has been produced from the data stored as noise data.**

4.8 Background Images

We created the CSV files for background images by using the LassoSelector tool to mark the point clusters that did not belong to any of the classes. The original amount of background images was 162 which we then increased by doing image transforms and the final count of background images became 984. The background images were passed with the training dataset to the YOLOv8 model. These images are important to reduce the probability of false positives [48]. Figure 4.6 is an example of a background image used during training.

Chapter 5

Input Modelling

In this thesis project, one of the major contributions is how to convert the point data of the radar into a format that YOLOv8 will understand and process. YOLOv8 accepts images and videos as input and therefore, we explored strategies to convert a set of point data into an image.

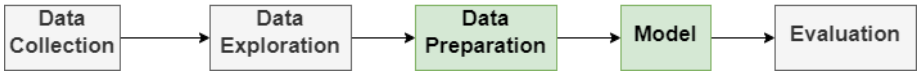


Figure 5.1: **The highlighted blocks, Data Preparation, and Model Exploration are discussed in this chapter.**

We covered the step-by-step details of transforming the point cloud into an image. This covers the Data Preparation and Model blocks of our DL application pipeline.

5.1 Setting the Context

Usually, any image can be represented as a set of arrays, each array representing a channel. For example, an image may have Red, Green, and Blue (RGB) channels. In this case, each channel value contains the contribution of the specific color across different pixels in the image. Each cell of the array represents a pixel which holds the quantitative value of the colour representing the array. So as shown in figure 5.2, an image is mainly an array containing information on the color of the locations. But in the case of the radar output, as shown in the table on the right of the same figure, it contains a collection of properties that describe the state of the target. There is no direct relation between the point cloud format with the image format.

In general, the image processing algorithm extracts the features like edges, contours, size, aspect ratio, etc of the object to be detected in the image. On the contrary, a radar's output contains the state information of the target in the form of its range, velocity, SNR, etc. Hence, we analyzed the various characteristics of an image and composed ways to transform the point cloud data into

that format.

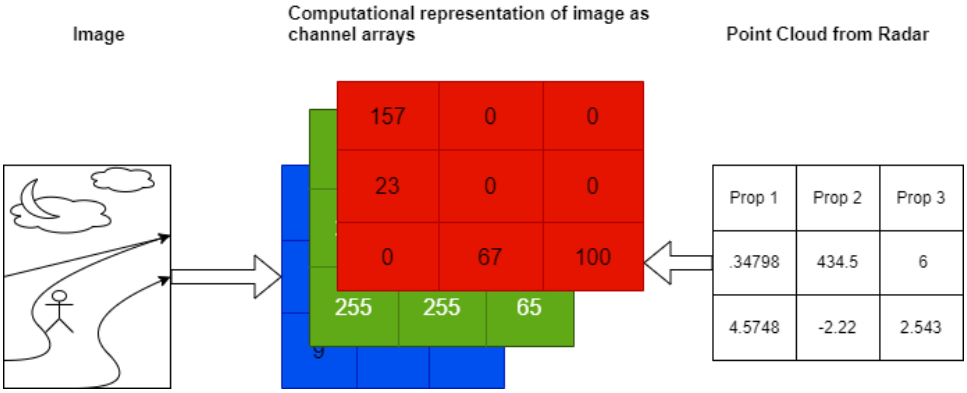


Figure 5.2: **The image is represented as a series of arrays for computational purposes. Our goal is to convert the point cloud as shown in the extreme right into a series of arrays resembling the format of the image.**

The existing work of using YOLO model for mm-wave radar output has used heat maps of the various target state properties[22]. But in this thesis, we explored this image formation differently. The target object is visually represented by its point cluster. Equivalent to the Red, Green, and Blue Channels of a conventional image, we used the state properties of the target as channels. Hence the color value of the pixel becomes equivalent to the value of the property representing the channel.

5.2 Overview of Point Cloud to Image Conversion

As the first step of the formatting, we created the array structure by deriving its shape and size depending on the radar frame depth and width. Then we devised a way to map the state property values into the array that would be equivalent to pixel values as in the case of images.

We faced one problem while doing this array formation and mapping. Multiple points were mapped to the same pixel location leading to information loss because a pixel can be represented only by a single value. We addressed this issue after we set the image focus.

Once the skeleton was established, the next challenge was to determine the focus of the image - whether to zoom into the cluster and learn its features or to investigate the same in its original position in the scene. This was important because, depending on the focus, the strength of feature extraction was dependent.

Finally we experimented with the properties we obtained from the radar output

to decide the optimum ones to be used as channels. In the following sections, we discuss in detail the steps to actualize the strategy.

It is important to highlight that we discussed object focus in this report before array size and shape discussion, because of being motivated by the fact that depending on the focus, the scaling and mapping of the points were done into the channel array. Also, the bounding box calculation for annotation depends on the focus. Hence, though it might look counter-intuitive to discuss object focus before discussing the array formatting and shape, it is important to understand the focus concepts as they have been referred to frequently throughout the rest of the chapter.

5.2.1 Object of Focus

It is crucial to determine the content and focus of the image. While creating an image out of the point cloud, we only have the positional coordinates of the points and their state properties. We thought of two ways in which the target object could be represented. One was to make the point cluster area as the entire image. We refer to such images as the Zoomed Image. The other was to keep the cluster in the original scene and remove the remaining clusters. This type of image is termed as a Full Image in the rest of the thesis report. Figure 5.4 pictorially demonstrates the same.

The advantage of the Full Image is along with the cluster information, the position of the cluster is also considered. This is important because the range is one of the properties that influence the shape and size of the target cluster. Hence providing the location enhances the contextual information for detection.

We calculated the bounding boxes for the images in the two approaches in the following respective methods -

- **Zoomed Image** - The bounding box of the Zoomed Image is equal to the entire image. Hence the bounding box annotation becomes as [box_x_centre = 0.5, box_y_centre = 0.5, box_width = 1.0, box_height = 1.0]. It should be noted that they are all in normalized form with respect to the dimension of the image.

- **Full Image** - The cluster's height, width, and x,y center coordinates were calculated and then normalized by the dimension of the input image. This is different from the Zoomed Image as its values vary depending on the actual shape and size of the point cluster and also its position in the original scene.

5.2.2 Image Array Format

The radar output contains the range and azimuth values of every point. From these, we extracted the x and y values which we used to derive the arrays of an image channel. In this project, the range of x is from -10 to 5 while that of y is from 0 to 12. Hence, we first translated the values into the positive quadrant of the coordinate system and then scaled it to make the data spread out optimally

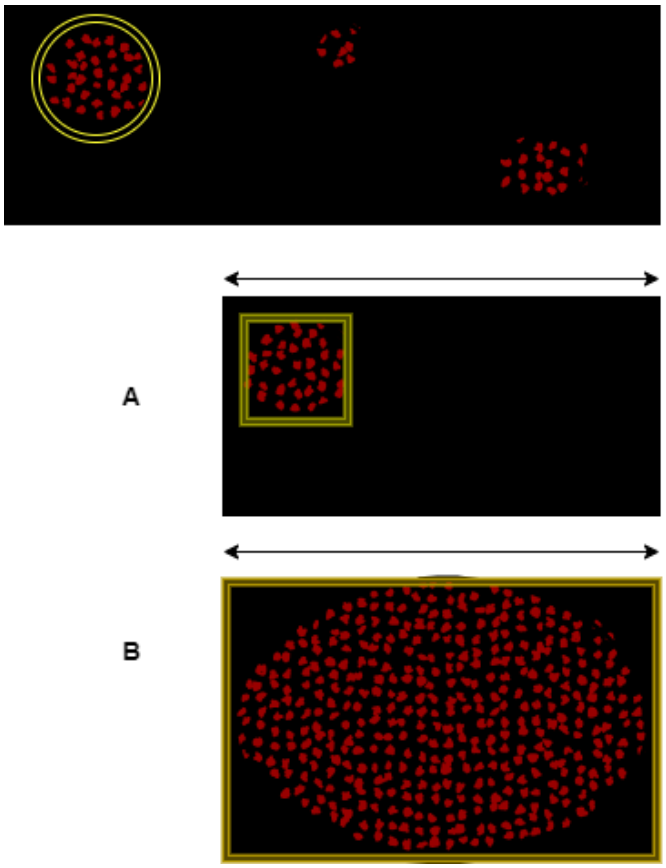


Figure 5.3: **The cluster highlighted with yellow is used to create a Full Image referred to as A and a Zoomed Image referred to as B.**

in the channel array.

To translate a Full Image to the positive quadrant we subtracted -10 from all the x values. For the Zoomed Image, we subtracted the min x value present in the point cloud from all the x values of the same. For the scaling factor, we did trial and error with 10, 32, and 31. The 31 value was finalized as it did not create any out-of-bound index error during the mapping.

Next we converted the float values into integers to transform them into indices. We experimented with both the ceil and floor values, and we found the former produces `array index out of bound errors for some location`. Hence, the indices are created by taking the floor value to obtain an integer. By doing this all the points present in the point cloud are represented in an array format as shown in figure 5.5.
An example of the case of a Zoomed Image has been presented below.

1. As shown in figure 5.5, `x` has values of .234, 2.465....,-2.567. Let us assume that the lowest value of x in the list is -2.567.

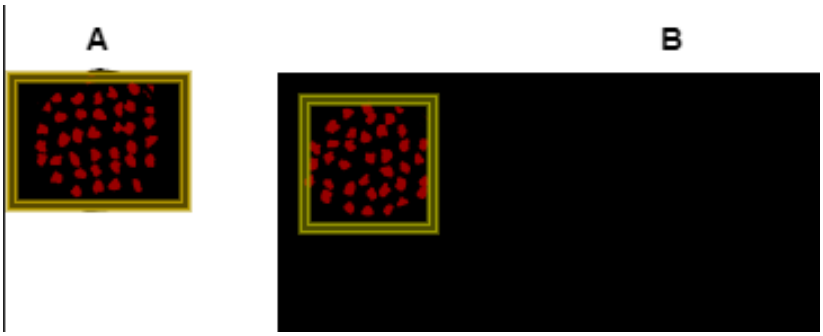


Figure 5.4: **The Zoomed Image (A) has the entire image as the object and hence has the bounding box equivalent to the image box. The Full Image (B) has its bounding box around the target inside the image.**

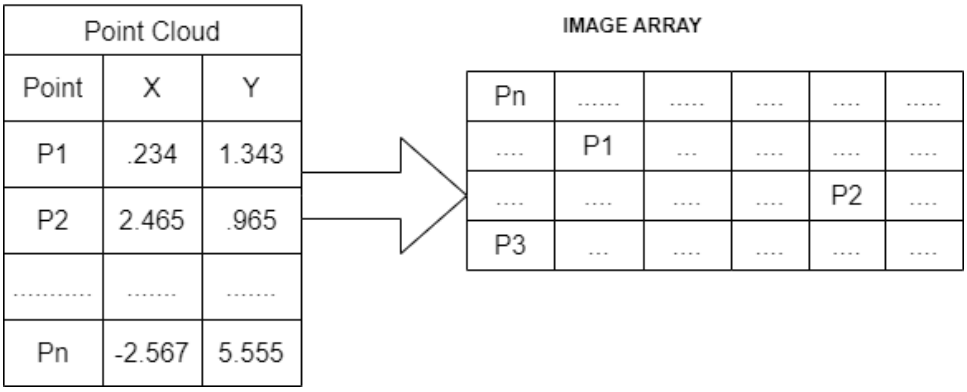


Figure 5.5: **The figure shows the points being mapped to their designated locations in the array based on their x & y coordinate values.**

2. Subtracting -2.567 to each x value converts them to 2.801, 5.032,.....,0.
3. Scaling all the x and y values by multiplying by 31 leads [(2.801, 1.343),(5.032, .965),....(0, 5.555)] to [(86.831, 41.633),(155.992, 29.915),....(0, 172.205)].
4. Taking the floor value of all the indices leads to [(86, 41), (155, 29),....., (0, 172)].

5.2.3 Image Size Selection

Every image is computationally represented as a multi-dimensional array and hence the height and width of the image is nothing but the number of rows and columns of the array. The standard image size of YOLOv8 is 640. This implies that either the height or the width should be of 640 pixels, making it easier to maintain the aspect ratio of the scene.

We experimented with 640 and 512 pixels as image sizes in two consecutive

Channel Combinations
SNR, Doppler, and Azimuth
Range, Doppler, and Azimuth
Std. Dev SNR, Std. Dev Doppler, Mean Azimuth
SNR, Doppler, Point Density

Table 5.1: **Various Combinations of the Target State Properties used as Image Channels.**

experiments to understand the effect of image size on accuracy. We maintained the aspect ratio of the original scene while mapping it to the desired image sizes.

Additionally, there is one requirement of YOLOv8 by Ultralytics and that is to have the image height and width to be divisible by 32. This is because the maximum stride during feature extraction in the backbone is 32. To be confident to have fulfilled this requirement, we multiplied the height and width of the image by 32 as shown in the equations 5.1 & 5.2 where height and width are that of the image. We add 31 to the existing width and height and then perform integer division on it followed by multiplying by 32. This ensures the two values remain divisible by 32.

$$width = int((width + 31)//32) * 32 \tag{5.1}$$

$$height = int((height + 31)//32) * 32 \tag{5.2}$$

For example, suppose an image has a size of 640x512. If we implement the two equations then it becomes ((640+31)//32) * 32 which is 640 and ((512 + 31)//32) * 32 which is 512. Now suppose the image has a size of 640x513. Then this calculation for 513 will make the new width as ((513+31)//32) * 32 which is 544. So now the new size of the image becomes 640x544.

5.2.4 Image Channels

Images consist of one or more channels which are represented as arrays. In general, images are of the Red, Green, and Blue channels interchangeably called the RGB channels. The image is represented based on the values of the pixels in these channel arrays. A pixel is one cell of the channel array.

Drawing an analogy to the RGB channels, in this project we drew equivalence to them to the various state properties of the target as shown in Figure 5.6. Similar to how the values of each channel color are mapped to the pixels of the channel arrays, we mapped the state property values to the channel arrays. The most important four properties of the target measured by the radar are the SNR, Azimuth, Doppler, and Range. Using these properties, the following channel combinations as shown in the table 5.1 were proposed to identify the optimum combination. We used these combinations for both the Zoomed and the Full images.

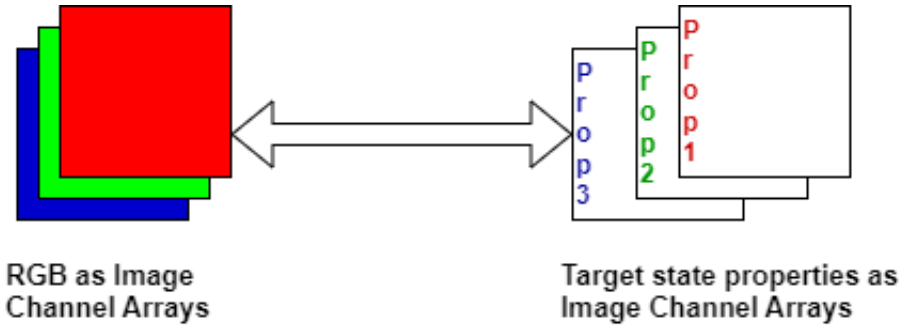


Figure 5.6: **The figure shows the RGB channel arrays of an image and equivalent channel arrays.**

5.2.5 Multiple-point Information Retention

We observed during the conversion of the x and y coordinates of the points into integers for array indices that it led to multiple points corresponding to the same row and column index values. Figure 5.7 demonstrates the issue where five points are mapped to (39,33) and the same was for (29,38). But one pixel can represent a single value only.

This posed a challenge since the total number of points in the radar output representing a target was already low owing to the low resolution. With such multiple points being tagged to the same cell of the array eventually led to a further decrease of the number of potential points to form the image of the point cluster. Thus we required a strategy to retain as much information as possible from such pixels.

To tackle this problem, we investigated various quantitative approaches to calculate one number that could represent all the points in the same cell location in the channel array.

Mean

It is the average value of all the points tagged to the same location of the channel array. For example, if the Doppler values are [0.3, 1.45, 0.2, 0.28] for the array location (0, 1) then the representative value would be 0.5575.

Inter Quartile Range (IQR)

A preliminary assessment of the radar output suggested that the Doppler and SNR values were abrupt in their range of values. To tackle this, we experimented with the interquartile range of the set of values by calculating the values for the third quartile and first quartile and then subtracting them to find the interquartile range. The percentile function [32] of the Numpy module of Python has been used to calculate the same.

class	serial	pixel	x1	y1	range	azimuth	snr	doppler
0	2	(39, 33)	-1.33651	8.140213	1.03115	0.139487	6.125	0
0	2	(39, 33)	-1.33651	8.140213	1.03115	0.139487	7.5	0.37424
0	2	(39, 33)	-1.33651	8.140213	1.03115	0.139487	23.875	0
0	2	(39, 33)	-1.33651	8.140213	1.03115	0.139487	73.25	0.37424
0	2	(39, 33)	-1.33651	8.140213	1.03115	0.139487	13.5	1.12272
0	2	(29, 38)	-1.19006	7.815398	0.687433	0.151111	13.5	0.124747
0	2	(29, 38)	-1.19006	7.815398	0.687433	0.151111	36	0.37424
0	2	(29, 38)	-1.19006	7.815398	0.687433	0.151111	37.125	0
0	2	(29, 38)	-1.19006	7.815398	0.687433	0.151111	75	0.37424
0	2	(29, 38)	-1.19006	7.815398	0.687433	0.151111	9.125	1.12272

Figure 5.7: **The figure shows a small excerpt from the details of a point cloud. It could be seen that the same location in the array holds many points with varying Doppler and SNR values**

The high variability of the SNR and Doppler in the dataset motivated this approach. For example, if the Doppler values are [0.3, 1.45, 0.2, 0.28] for the array location (0, 1) then the representative value would be 0.635.

Values related to the Highest SNR

The higher the value of SNR, the more reliable the signal information. Hence in this approach, we selected SNR as a metric to select the best point as the representative and then used the property values associated with that point.

Instead of using those values directly, we created a scale factor and multiplied the highest value with it. This factor i.e., (highest_value by max_value) either scales up or scales down the original value depending on whether the original value is the biggest in the list or there is some other value in the list that might have a higher value but did not get represented.

For example, the points in the location (39, 33) from the figure 5.7, the highest SNR value is 73.25. Hence, we consider the range, Doppler, azimuth, and SNR values associated with 73.25. Then the value for Doppler would be that (0.37424 * (0.37424/ 1.2272)) i.e. 0.1141. This logic was applied to Doppler and Azimuth values. In case the max values become zero in any scenario, we have not scaled the value and multiplied the highest value by 1.

5.3 YOLOv8 and Tensors

YOLOv8 by Ultralytics supports three-channel images to date. Hence we could try only a combination of the three properties that we got from the radar output. But there are more properties available and sending more information about the target might improve the detection accuracy of the model.

Thus, in this experiment, we increased the number of channels from three to four. Though images with four channels exist, the main motivation to use NPY

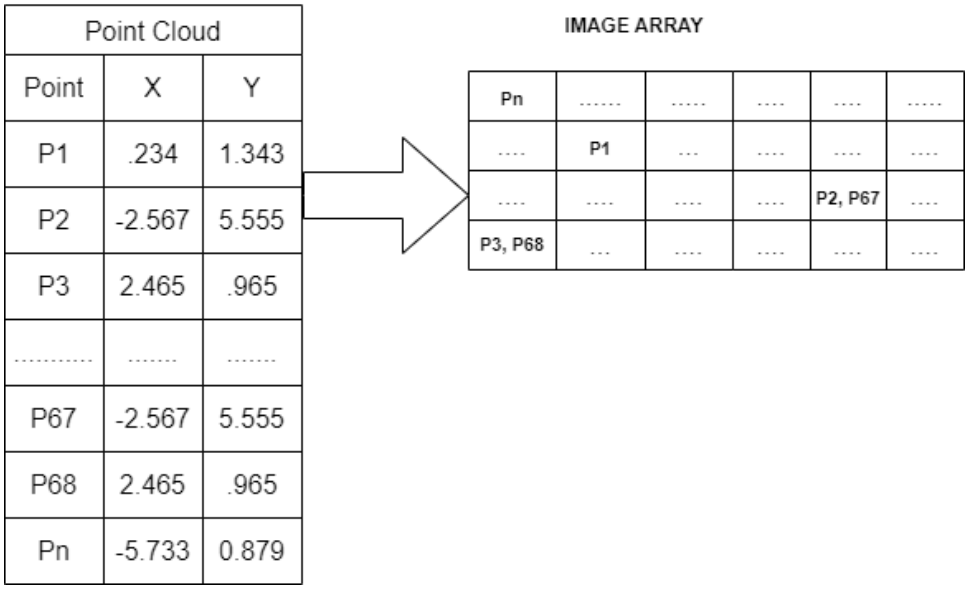


Figure 5.8: **The figure shows many points being mapped to the same cell of the channel array.**

format files was that they could be increased later to more channels. We selected the large version of the YOLOv8 model of Ultralytics which has 365 layers so that it can accommodate an increase in the number of input channels.

We customized the existing code of the YOLOv8 model to allow the processing of NPY files with four channels.

5.4 Baseline Experiments

Though we worked entirely with the YOLOv8 model, as our baseline experiments we also implemented the PyTorch version of the PointNet model on the radar data.

We converted the same radar data that we received from the AMS Institute into Object File Format (OFF) files and then trained them on a PyTorch implementation of the PointNet[31]. Even though we selected YOLOv8 as our selected model, we wanted to experiment with a model dedicated to point cloud.

5.5 List of Experiments

Table 5.2 documents the comprehensive list of experiments conducted and explained later in Chapter 6 for analysis and evaluation.
The relevant code files created and used during this thesis project are kept in the link `https://github.com/bagchim/BagchiThesis_PCmmWave`.

Object to Focus	Channels	ImgSz	Info Retention
Zoom	Red, Blue, Green	640	Not Applicable
Full	Red, Blue, Green	640	Not Applicable
Zoom	SNR, Doppler, Azimuth	640	Mean
Zoom	SNR, Doppler, Azimuth	640	IQR
Zoom	SNR, Doppler, Azimuth	512	IQR
Zoom	SNR, Doppler, Azimuth	640	Highest SNR
Full	Range, Doppler, Azimuth	640	IQR
Full	SNR, Doppler, Azimuth	640	IQR
Full	SNR, Doppler, Point Density	640	Highest SNR
Full	Std. Dev SNR, Std. Dev Doppler, Mean Azimuth	640	IQR
Full	Std. Dev SNR, Std. Dev Doppler, Mean Azimuth	512	IQR
Full	SNR, Doppler, Azimuth, Point Density	512	Highest SNR
Full	SNR, Doppler, Azimuth, Point Density	640	Highest SNR

Table 5.2: **The following experiments were conducted during the thesis project showing their image size, information retention strategy, and channel combination.**

Chapter 6

Evaluation

In this chapter, we discuss the results of all those experiment results that were conducted and make inferences.

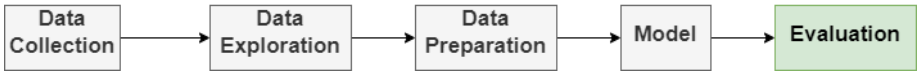


Figure 6.1: **The highlighted block Evaluation is discussed in this chapter.**

6.1 Evaluation Metrics

Before getting into the in-depth analysis, first, we introduce the evaluation metrics that are important in determining the performance of a model.
In object detection, the important metrics for evaluating an application model are its accuracy (eqtn. 6.1) and mean average precision (mAP) value (eqtn. 6.2) of the Intersection Over Union ratio (eqtn. 6.3) of 0.5.

$$\text{Accuracy} = \frac{\text{True Positive} + \text{True Negative}}{\text{Total Predictions}} \tag{6.1}$$

$$\text{mAPInference} = \frac{\sum_{i=1}^{N} \text{AP}_i}{N} \tag{6.2}$$

$$IoU = \frac{\text{Area of Intersection}}{\text{Area of Union}} \tag{6.3}$$

Our work was to classify the point cluster from the radar output as one of the six classes on which the YOLOv8 model has been trained. The shape and size of the point cluster of a target varies with its position relative to the radar. Thus, the benefit of using an object detection model for the point cloud output of the radar is it provides additional information on the location of the target.

Class	% of instances above 500 points
0	.28
1	2.54
2	5.17
3	17.03
4	35.04
5	12.69

Table 6.1: **The percentage of instances in each class where the point cluster has above 500 points.**

Thus in this project, accuracy is the most important evaluation metric. Additionally, the precision and recall values have also been observed since the misclassification of the cluster can lead to an erroneous total count of the people. Hence in all the cases, the analysis was primarily based on accuracy, precision, and recall.
The details of the experiments have been consolidated in the table 6.4 and should be referred to know about the various parameter values set to conduct the experiments.

6.2 Baseline Results

At the start, we evaluated the YOLOv8 model by creating the input images with only the positional data of the reflected radar points. It determined how well the clusters can be detected based on their shape & size. This was done for both the types of input, i.e. the zoomed image and the full image, to investigate the effect of location information.

The zoomed image gave an accuracy of 0.6466 which would be translated to 64.6% and the full image had an accuracy of .6826, translated to 68.26%. Hence with no information of the target being passed apart from its position and shape, YOLOv8 could classify the clusters with a decent accuracy. After setting the foundation, we formulated the later experiments to improve the classification accuracy for the clusters by implementing the target's state properties as measured using the mm-wave radar.

In addition to this, we implemented the PyTorch implementation of the PointNet model [31]. The model gave an accuracy of 31.5% after training on 50 epochs with a batch size of 32. Now this could be due to a couple of reasons. Firstly, the mm-wave radar output is very scarce. From the data exploration results, we found the below results for point density. We calculated the count of instances with over 500 points and the class-wise results are as follows in table 6.1.

It was clear from the graph 6.2 that the mm-wave radar data was very scarce. The baseline result of YOLOv8 was 32.9% higher than that of the PointNet implementation that we used. Also unlike YOLOv8, the PointNet implementation cannot extract features at various scales. The multi-scalar feature extraction

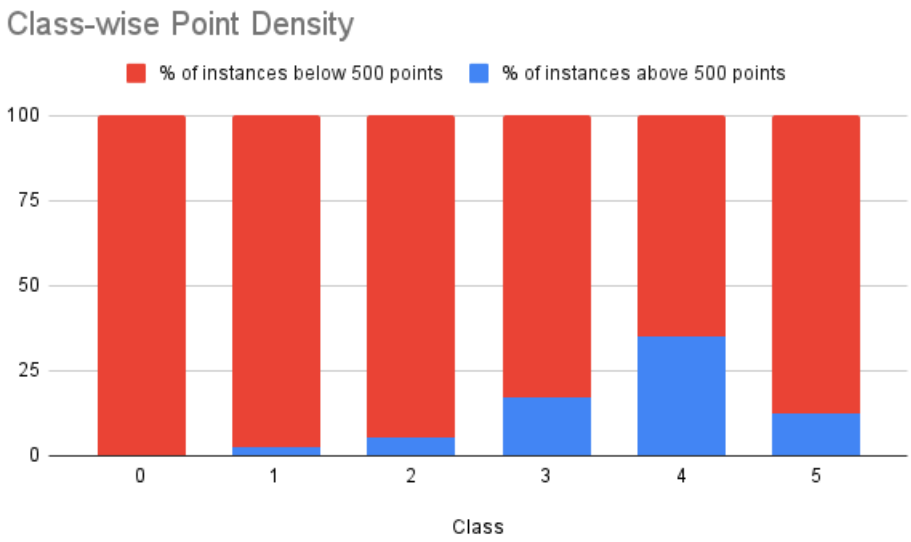


Figure 6.2: **The percentage of number of instances present per class where the point count is above 500 points.**

characteristic of YOLOv8 makes it a good choice for such a dynamic application.

6.3 Object to Focus

The first aspect of modelling the input was to decide on the focus of the cluster in the image. The experiments were conducted by creating the input for zoomed image and full image keeping all other factors constant.

As observed in figure 6.3, the model where the full image along with its location information performed better than where only the cluster was sent. This is due to the following reasons fact

- In the case of the full image the location information of the cluster is inherent in the original scene which helps the algorithm to have a sense of the size of the object relative to its distance from the data-capturing device.
- The point cloud being of low resolution, the input image did not have any significant advantage of implementing a zoomed image. Moreover, YOLOv8 has a pyramid structure for feature extraction to extract features from global to local scale. Hence, it is robust to low-resolution images.
- This low-resolution data of mm-wave radar requires lesser computational memory and hence makes detection fast in the object detection model.

Thus for the final model, we selected the full image to be our approach to focus on the target cluster.

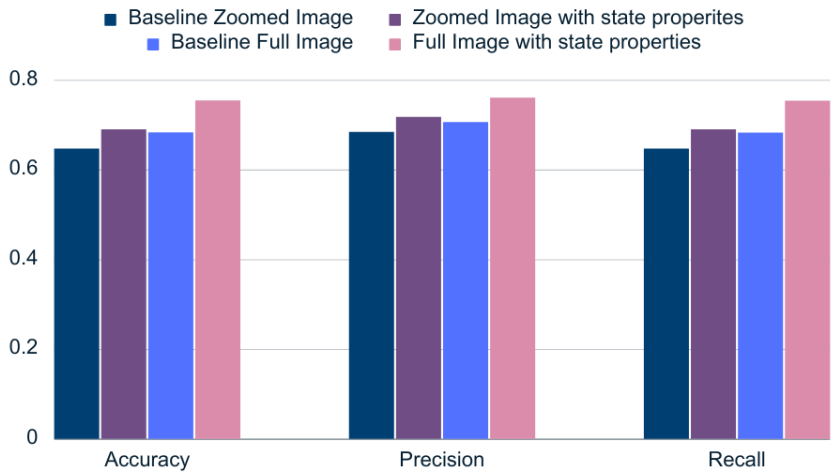


Figure 6.3: **The accuracy, precision, and recall for Zoomed and Full images have been compared.**

6.3.1 Image Size

We conducted four sets of experiments whose results have been recorded in table 6.2. It could be seen that there was a slight difference in the inference accuracy of the zoomed and the full images when their image sizes were reduced.
The invariant nature of the accuracy towards image size for the sizes 640 and 512, could be because mm-wave data is already widely spaced and also less in number. Since there was no significant change by varying the image sizes, we kept it to the standard YOLOv8 of 640 pixels in width.

6.3.2 Multiple point Information Retention

In Chapter 5, we discussed three strategies to select the most optimal one to reduce the loss of information when multiple points have the same cell location in the channel array of the image. The experiments to compare the information retention strategies were conducted on zoomed images. The channels were SNR, Doppler, and Azimuth. The results of the three experiments were evaluated and recorded in 6.3.

The results show that the strategy of the highest SNR proved to be the most efficient one. The results indicate that this is because of the following points.

1. The mean value of a highly spread-out dataset would never correctly represent the entire set. Hence the experiment with the mean value secured the lowest accuracy amongst the three.

Image Focus	Image Size	Accuracy (%)	Precision (%)
Zoomed	640	68.98	71.72
Zoomed	512	67.98	69.43
Full	640	70.55	71.97
Full	512	71.2	72.06

Table 6.2: **The accuracy and precision of images with sizes 640 and 512 respectively have been compared.**

Retention Strategy	Accuracy (%)	Precision (%)
Interquartile Range	68.98	71.72
Mean	67.96	70.56
Highest SNR	70.09	71.38

Table 6.3: **The accuracy, precision, and recall values of the different information retention strategies.**

2. It was observed that the remaining two strategies led to close values of accuracy. If we analyze the results of the IQR strategy, it represents the range of the values in the case of the highly varied datasets. In this case, the result performed better because most of the pixels had multiple points mapped to them. Which led to the majority of the pixels having a range to represent. Moreover, when the data was analyzed before performing these experiments, we found that the standard deviation of the SNR in particular was very high. However, this analysis was constrained to the particular dataset that we created. Hence, we can conclude that if the current dataset is increased in the future with more instances then it is essential to analyze the variability of the SNR and Doppler values before choosing the IQR strategy.

3. The heuristics that we proposed involving the highest SNR resulted in the best accuracy among the three. This is mainly because the higher the value of SNR, the more credible the information that the reflected wave carries. Even though it might appear in some instances that a Doppler, Range, or Azimuth value has been selected that is lower than its highest, we can have assurance that the one having the highest signal strength has been chosen. Also, to take care of the same, we use the scaling factor so that the values are optimized to represent the overall set.

Hence, for the final model, the highest SNR strategy was selected for information retention.

6.3.3 Channels

Different combinations of the three channels of an image were experimented to explore which properties demonstrate the characteristics of the cluster that help in its detection. The table 6.4 shows how the model performed on the various inputs.
The results of the combination of SNR-Doppler-Azimuth and SNR-Doppler-Point Density were almost similar. The least performing combination was that

Object to Focus	Channels	ImgSz	Info Retention	Accuracy(%)	Precision (%)
Zoom	Red, Blue, Green	640	Not Applicable	64.66	68.41
Full	Red, Blue, Green	640	Not Applicable	68.26	70.56
Zoom	SNR, Doppler, Azimuth	640	Mean	67.96	70.56
Zoom	SNR, Doppler, Azimuth	640	IQR	68.98	71.72
Zoom	SNR, Doppler, Azimuth	512	IQR	67.98	69.43
Zoom	SNR, Doppler, Azimuth	640	Highest SNR	70.09	71.38
Full	Range, Doppler, Azimuth	640	IQR	68.31	71.2
Full	SNR, Doppler, Azimuth	640	IQR	75.26	76.06
Full	SNR, Doppler, Point Density	640	Highest SNR	72.83	74.73
Full	Std. Dev SNR, Std. Dev Doppler, Mean Azimuth	640	IQR	70.55	71.97
Full	Std. Dev SNR, Std. Dev Doppler, Mean Azimuth	512	IQR	71.2	72.06
Full	SNR, Doppler, Azimuth, Point Density	512	Highest SNR	79.66	80.21
Full	SNR, Doppler, Azimuth, Point Density	640	Highest SNR	79.21	80.31

Table 6.4: **The following experiments were conducted during the thesis project showing their image size, information retention strategy, and channel combination along with the accuracy and precision of the model.**

of Range-Doppler-Azimuth. This might be because the range and azimuth are properties that contribute to the positional information of the points and hence the shape. The fact that the Range-Doppler-Azimuth combination was not as accurate as the SNR-Doppler-Azimuth value indicates that sending the Range information might have been redundant data. Hence we used the combination of SNR, Doppler, Azimuth, and Point Density while creating the four-channel tensor.

6.3.4 Tensor vs Image

The main motivation to modify the input format as a tensor was for the implementation of increasing the number of channels. We extended the model to include 4 channels instead of 3. The motivation was to investigate whether the YOLOv8 model could handle additional channels. The large scale of the model was used because the first convolution layer of the architecture had 64 channels as output, unlike the other scales like nano, small, or medium.

There are a few highlights of using tensors over images in an object detection model from the CV field.

- Firstly, images were lightweight in file size compared to tensor files. That made training faster compared to training tensors.
- Tensors paved the way to increase the number of channels. The characteristics of YOLOv8 can be maintained by changing the architecture of the model to incorporate the increase in the number of channels.
- We could not use the YOLOv8 image transforms on the tensors. But we overcame this challenge by incorporating transforms suitable for tensors.

6.4 Comparison of the models

We observed from figure 6.5 that starting from a baseline accuracy value of 64.66%, the results of the model improved by formatting the radar output data.

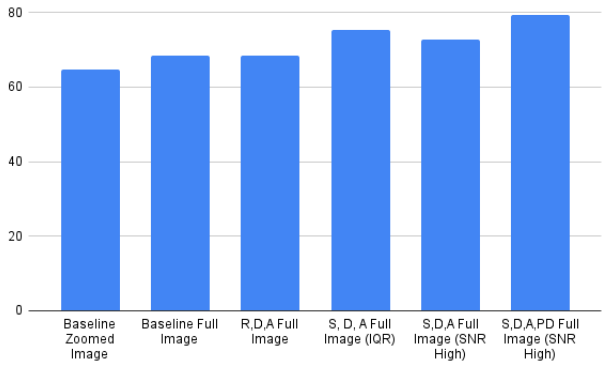


Figure 6.4: **The accuracy of the models implemented was compared. Here S,D,A stands for SNR, Doppler, and Azimuth, and R,D,A stands for Range, Doppler, and Azimuth.**

The final formatted input model secures an accuracy of 79.21%. After researching its various aspects from focus, information retention, resolution, and channel combination to the input type being tensor or image, it was seen that the highest-performing input was a combination of SNR, Doppler, Azimuth, and Point Density. The image size has been maintained to be 640 and the focus was the cluster along with its original position in the frame. It was also clearly seen that introducing the target's state properties along with its shape and size did play a significant role in improving the detection accuracy.

6.5 Final Model

Below are the details for the final model.

- Image Size - 640
- Input Type - Four-Channel NPY file
- Channels - SNR, Doppler, Azimuth and Point Density
- Information Retention Strategy - Highest SNR
- Object to Focus - Full Image
- YOLOv8 Model - Customized Large scale YOLOv8
- Accuracy - 79.215
- Precision - 80.316
- Recall - 79.21
- inference iOu - 0.6
- mAP@0.5-val - .931

In addition to this, figure 6.5 shows the confusion matrix of the final model.

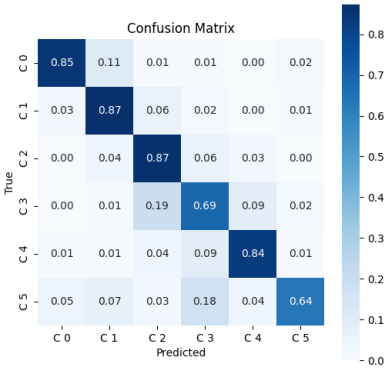


Figure 6.5: **The confusion matrix of the final model. C0 - C5 denoted the classes 0, 1, 2, 3, 4 and 5.**

Chapter 7

Conclusions

7.1 Conclusions

In this thesis, we developed a strategy to identify the strength of a group of people walking outdoors, from mm-wave radar output. The work concluded the following points.

- We created a dataset of over 20000 instances, with real-life radar and camera data of people walking in groups on the street, as provided by the AMS Institute team. The dataset contains CSV files of the point cloud of each group categorized into strengths of one, two three, and four. The dataset also contains the point cloud of groups greater than four people, all accumulated under the strength of five, as well as that of bikes, and background clutter. We refined the point cloud further using DBSCAN to remove additional outliers.

- We selected a computer vision algorithm YOLOv8 by Ultralytics for the detection model through the literature survey. YOLOv8 was preferred over other DL models so that a balance is maintained between the speed of detection and the accuracy. Furthermore, computer vision models for object detection usually work well with low-resolution images. Since the output from the mm-wave radar is of low resolution, we investigated if YOLOv8 can be a feasible choice for operation.

- We bridged the gap between the non-visual radar point cloud and the visual input format of YOLOv8 by drafting a method to utilize the target's state properties and visually present them as point cluster images. These point cluster images are then used as input to the YOLOv8 model. To do this, we used the x & y coordinates of the points to create an array format, the properties like azimuth, SNR, etc, to pass as channels and then developed the images.

- We extended the model by increasing its number of channels from three to four. We customized the YOLOv8 existing code to suit the processing of four-channel tensors. Starting from the baseline experiments that showed an accuracy of 64.66%, we finally achieved an accuracy of 79.21% by implementing four-channel tensors.

7.2 Future Work

Using mm-wave radar data in PC is a very vast research area and has several aspects that can be individually taken up as proper research topics. Through this thesis project, the problem of identifying the group strength of people outdoors has been tried to be addressed. While doing so, a computer vision algorithm was chosen to research its feasibility of being used for radar data. The motivation was that CV algorithms have decent speed and accuracy in object detection.

This work can be further extended in a couple of directions.

- The strength of the group our model can detect is up to four. Anything beyond four was categorized into five for simplicity. One major work can be to create a big dataset that covers groups of more than four people together, different outdoor locations, cluster information of vehicles, etc. In addition to that data should be collected at various times of the day and weather conditions to incorporate the dynamic nature of the outdoors. This will help in further exploration of mm-wave data for PC.
- In this project, we have extended the YOLOv8 input to four channels. More channels can be explored to support further improvement of detection accuracy. The neural architecture can be expanded or improved to accommodate more channels
- The focus has been only on identifying the number of individuals in the group, but in the broader perspective it will be used for PC using a Kalman Filter. In such a situation, the temporal information of the target can enhance the counting accuracy. This can be done by combining LSTM.
- Another approach would be to use transformers in combination with YOLOv8 and utilize the concept of attention to further the contextual information of the various state property combinations.

Bibliography

[1] Hajar Abedi, Shenghang Luo, and George Shaker. On the Use of Low-Cost Radars and Machine Learning for In-Vehicle Passenger Monitoring. In *2020 IEEE 20th Topical Meeting on Silicon Monolithic Integrated Circuits in RF Systems (SiRF)*, pages 63–65. IEEE, 1 2020.

[2] Mohammad Al-Sa'd, Serkan Kiranyaz, Iftikhar Ahmad, Christian Sundell, Matti Vakkuri, and Moncef Gabbouj. A social distance estimation and crowd monitoring system for surveillance cameras. *Computer Visions and Pattern Recognition*, 22(2), January 2022.

[3] Albumentations. Transforms (augmentations.transforms). `https://albumentations.ai/docs/getting_started/transforms_and_targets/`, 2023. Last accessed: Oct. 10, 2023.

[4] Mostafa Alizadeh, Hajar Abedi, and George Shaker. Low-cost low-power in-vehicle occupant detection with mm-wave FMCW radar. In *2019 IEEE SENSORS*, pages 1–4. IEEE, 10 2019.

[5] Alexey Bochkovskiy, Chien-Yao Wang, and Hong-Yuan Mark Liao. Yolov4: Optimal speed and accuracy of object detection. *Computer Vision and Pattern Recognition*, 2020.

[6] Graham Brooker and Graham M Brooker. Understanding millimetre wave FMCW radars Radar Education View project Mining Radar View project Understanding Millimetre Wave FMCW Radars. Technical report.

[7] R. Qi Charles, Hao Su, Mo Kaichun, and Leonidas J. Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 77–85, 2017.

[8] Han Cui and Naim Dahnoun. High Precision Human Detection and Tracking Using Millimeter-Wave Radars. *IEEE Aerospace and Electronic Systems Magazine*, 36(1):22–32, 1 2021.

[9] Peixian Gong, Chunyu Wang, and Lihua Zhang. MMPoint-GNN: Graph Neural Network with Dynamic Edges for Human Activity Recognition through a Millimeter-Wave Radar. In *Proceedings of the International Joint Conference on Neural Networks*, volume 2021-July. Institute of Electrical and Electronics Engineers Inc., 7 2021.

[10] Mahmudul Hasan, Junichi Hanawa, Riku Goto, Ryota Suzuki, Hisato Fukuda, Yoshinori Kuno, and Yoshinori Kobayashi. Lidar-based detection, tracking, and property estimation: A contemporary review. *Neurocomputing*, 506:393–405, 2022.

[11] Kaiming He, Georgia Gkioxari, Piotr Dollar, and Ross Girshick. Mask R-CNN. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 2980–2988. IEEE, 10 2017.

[12] Xu Huang, Hasnain Cheena, Abin Thomas, and Joseph K. P. Tsoi. Indoor Detection and Tracking of People Using mmWave Sensor. *Journal of Sensors*, 2021:1–14, 2 2021.

[13] Texas Instruments. Iwr6843isk. `https://www.ti.com/tool/IWR6843ISK`, 2023. Last accessed: Oct. 10, 2023.

[14] Salma Kammoun Jarraya, Maha Hamdan Alotibi, and Manar Salamah Ali. A deep-cnn crowd counting model for enforcing social distancing during covid19 pandemic: Application to saudi arabia's public places. *Computers, Materials Continua*, 66(2), 2021.

[15] Glenn Jocher. Brief summary of yolov8 model structure 189. `https://github.com/ultralytics/ultralytics/issues/189`, 2023. Last accessed: Oct. 10, 2023.

[16] Glenn Jocher, Ayush Chaurasia, and Jing Qiu. Ultralytics yolov8 version 8.0.188. `https://github.com/ultralytics/ultralytics",orcid="0000-0001-5950-6979,0000-0002-7603-6750,0000-0003-3783-7069`, 2023. AGPL-3.0.

[17] Margrit Kasper-Eulaers, Nico Hahn, Stian Berger, Tom Sebulonsen, Øystein Myrland, and Per Egil Kummervold. Short communication: Detecting heavy goods vehicles in rest areas in winter conditions using yolov5.

[18] Abdullah Khalili, Abdel-Hamid Soliman, Md Asaduzzaman, and Alison Griffiths. Wi-Fi sensing: applications and challenges. *The Journal of Engineering*, 2020(3):87–97, 3 2020.

[19] Muhammad Asif Khan, Hamid Menouar, and Ridha Hamila. Revisiting crowd counting: State-of-the-art, trends, and future perspectives. *Image and Vision Computing*, 129:104597, 2023.

[20] Sylvia T. Kouyoumdjieva, Peter Danielis, and Gunnar Karlsson. Survey of Non-Image-Based Approaches for Counting People. *IEEE Communications Surveys and Tutorials*, 22(2):1305–1336, 4 2020.

[21] Mohamed Lamane, Mohamed Tabaa, and Abdessamad Klilou. Classification of targets detected by mmWave radar using YOLOv5. In *Procedia Computer Science*, volume 203, pages 426–431. Elsevier B.V., 2022.

[22] Mohamed Lamane, Mohamed Tabaa, and Abdessamad Klilou. Classification of targets detected by mmwave radar using yolov5. *Procedia Comput. Sci.*, 203(C):426–431, jan 2022.

[23] B. Leibe, E. Seemann, and B. Schiele. Pedestrian detection in crowded scenes. 2005.

[24] Shenglei Li and Reiko Hishiyama. Counting and tracking people with mmWave sensor in the restaurant. Technical report, 2022.

[25] Y. Li, X. Zhang, and D. Chen. Csrnet: Dilated convolutional neural networks for understanding the highly congested scenes. *EEE/CVF Conference on Computer Vision and Pattern Recognition*, November 2018.

[26] Tsung-Yi Lin, Piotr Doll, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. *IEEE*, 2016.

[27] Zhe Lin and Larry S. Davis. Shape-based human detection and segmentation via hierarchical part-template matching. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2010.

[28] Zongquan Ling, Weinan Zhou, Yuhao Ren, Jiping Wang, and Liquan Guo. Non-contact heart rate monitoring based on millimeter wave radar. *IEEE Access*, 10:74033–74044, 2022.

[29] Chen Change Loy, Ke Chen, Shaogang Gong, and Tao Xiang. Crowd Counting and Profiling: Methodology and Evaluation. pages 347–382. 2013.

[30] matplotlib. Lasso selector. `https://matplotlib.org/stable/gallery/widgets/lasso_selector_demo_sgskip.html`, 2023. Last accessed: Oct. 09, 2023.

[31] Nikita Karaev and Irina Nikulina. Pytorch implementation of.

[32] NumPy. numpy.percentile. `https://numpy.org/doc/1.20/reference/generated/numpy.percentile.html`, 2023. Last accessed: Oct. 09, 2023.

[33] Sameera Palipana, Dariush Salami, Luis A. Leiva, and Stephan Sigg. Pantomime: Mid-air gesture recognition with sparse millimeter-wave radar point clouds. 5(1), mar 2021.

[34] Akarsh Prabhakara, Tao Jin, Arnav Das, Gantavya Bhatt, Lilly Kumari, Elahe Soltanaghai, Jeff Bilmes, Swarun Kumar, and Anthony Rowe. High resolution point clouds from mmwave radar. *High Resolution Point Clouds from mmWave Radar*, July 2023.

[35] Charles R. Qi, Li Yi, Hao Su, and Leonidas J. Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, page 5105–5114, Red Hook, NY, USA, 2017. Curran Associates Inc.

[36] Joseph Redmon and Ali Farhadi. Yolov3: An incremental improvement. 4 2018.

[37] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. 6 2015.

[38] Nicolas Scheiner, Florian Kraus, Nils Appenrodt, Jürgen Dickmann, and Bernhard Sick. Object detection for automotive radar point clouds – a comparison. *AI Perspectives*, 3(1):6, 11 2021.

[39] Ole Schumann, Markus Hahn, Jurgen Dickmann, and Christian Wohler. Semantic Segmentation on Radar Point Clouds. In *2018 21st International Conference on Information Fusion (FUSION)*, pages 2179–2186. IEEE, 7 2018.

[40] scikit learn. sklearn.cluster.dbscan. `https://scikit-learn.org/stable/modules/generated/dbscan-function.html`, 2023. Last accessed: Oct. 09, 2023.

[41] Lorenzo Servadei, Huawei Sun, Julius Ott, Michael Stephan, Souvik Hazra, Thomas Stadelmayer, Daniela Sanchez Lopera, Robert Wille, and Avik Santra. Label-Aware Ranked Loss for Robust People Counting Using Automotive In-Cabin Radar. In *ICASSP 2022 - 2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 3883–3887. IEEE, 5 2022.

[42] X. Shi, X. Li, C. Wu, S. Kong, J. Yang, and L. He. A real-time deep network for crowd counting. *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, November 2020.

[43] Daniel J. Solove. A taxonomy of privacy. *GWU Law School Public Law Research Paper No. 129*, 154(3), January 2006.

[44] Heemang Song, Youngkeun Yoo, and Hyun-Chool Shin. In-Vehicle Passenger Detection Using FMCW Radar. In *2021 International Conference on Information Networking (ICOIN)*, pages 644–647. IEEE, 1 2021.

[45] Juan Terven and Diana Cordova-Esparza. A comprehensive review of yolo: From yolov1 and beyond. 2023.

[46] Ultralytics. About the optimizer 3360. `https://github.com/ultralytics/ultralytics/issues/3360`, 2023. Last accessed: Oct. 09, 2023.

[47] Ultralytics. Brief summary of yolov8 model structure. `https://github.com/ultralytics/ultralytics/issues/189`, 2023. Last accessed: Oct. 09, 2023.

[48] Ultralytics. Tips for best training results. `https://docs.ultralytics.com/yolov5/tutorials/tips_for_best_training_results/#dataset`, 2023. Last accessed: Oct. 09, 2023.

[49] P. Viola and M.J. Jones. Robust real-time face detection. *International Journal of Computer Vision*, 2004.

[50] Steve White. Cctv cameras by countries cities (2023 guide). `https://upcomingsecurity.co.uk/security-guides/cctv-camera-guides/cctv-by-country/`, 2023. Last accessed: Oct. 09, 2023.

[51] Wikipedia. Radar. `https://en.wikipedia.org/wiki/Radar#`, 2023. Last accessed: Oct. 09, 2023.

[52] Jiayu Wu, Zhanyu Zhu, and Haipeng Wang. Human Detection and Action Classification Based on Millimeter Wave Radar Point Cloud Imaging Technology. In *2021 Signal Processing Symposium (SPSympo)*, pages 294–299. IEEE, 9 2021.

[53] Yicheng Yao, Changyu Liu, Hao Zhang, Baiju Yan, Pu Jian, Peng Wang, Lidong Du, Xianxiang Chen, Baoshi Han, and Zhen Fang. Fall detection system using millimeter-wave radar based on neural network and information fusion. *IEEE Internet of Things Journal*, 9(21):21038–21050, 2022.

[54] C. Zhang, H. Li, X. Wang, and X. Yang. Cross-scene crowd counting via deep convolutional neural networks. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[55] Peijun Zhao, Chris Xiaoxuan Lu, Jianan Wang, Changhao Chen, Wei Wang, Niki Trigoni, and Andrew Markham. mID: Tracking and Identifying People with Millimeter Wave Radar. In *2019 15th International Conference on Distributed Computing in Sensor Systems (DCOSS)*, pages 33–40. IEEE, 5 2019.

[56] Peijun Zhao, Chris Xiaoxuan Lu, Jianan Wang, Changhao Chen, Wei Wang, Niki Trigoni, and Andrew Markham. Human tracking and identification through a millimeter wave radar. *Ad Hoc Networks*, 116:102475, 5 2021.

[57] Jing Zong, Binke Huang, Liang He, Bo Yang, and Xiaoyan Cheng. Device-Free Crowd Counting Based on the Phase Difference of Channel State Information. In *2020 IEEE International Conference on Information Technology,Big Data and Artificial Intelligence (ICIBA)*, pages 1343–1347. IEEE, 11 2020.

[58] C Álvarez-Aparicio, AM Guerrero-Higueras, FJ Rodríguez-Lera, Clavero J Ginés, Rico F Martín, and v Matellán. People detection and tracking using lidar sensors. *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 8(3), November 2019.

Appendix A

APPENDIX TITLE

A.1 Architecture Diagram of YOLOv8 by Ultralytics

The visualization of the figure A.1 was created by a Github user RangeKing.

A.2 Hyperparameter values and GPU Details

The hyperparameter values of the YOLOv8 model were kept constant for all the experiments unless mentioned explicitly. The following table A.1 shows the different values of the important hyperparameters.

YOLOv8 has the option to automatically select a suitable optimizer depending on the iterations[46]. If the number of iterations is less than 10,000 then AdamW optimizer is used. Otherwise, the Stochastic Gradient Descent (SGD) optimizer has been used. For the experiments conducted during this research, the SGD has been used.

Hyperparameter	Value
Epochs	100
image size	640
learning rate	0.01
iou	0.7
batch	16
momentum	0.9

Table A.1: **Hyperparameter values of YOLOv8s used for the experiments.**

In order to train the experiment models, the GPU server of the Embedded Network Systems department has been used. The details of the processor as used by YOLOv8 was - CUD: 0 (NVIDIA A40, 45449 MiB).

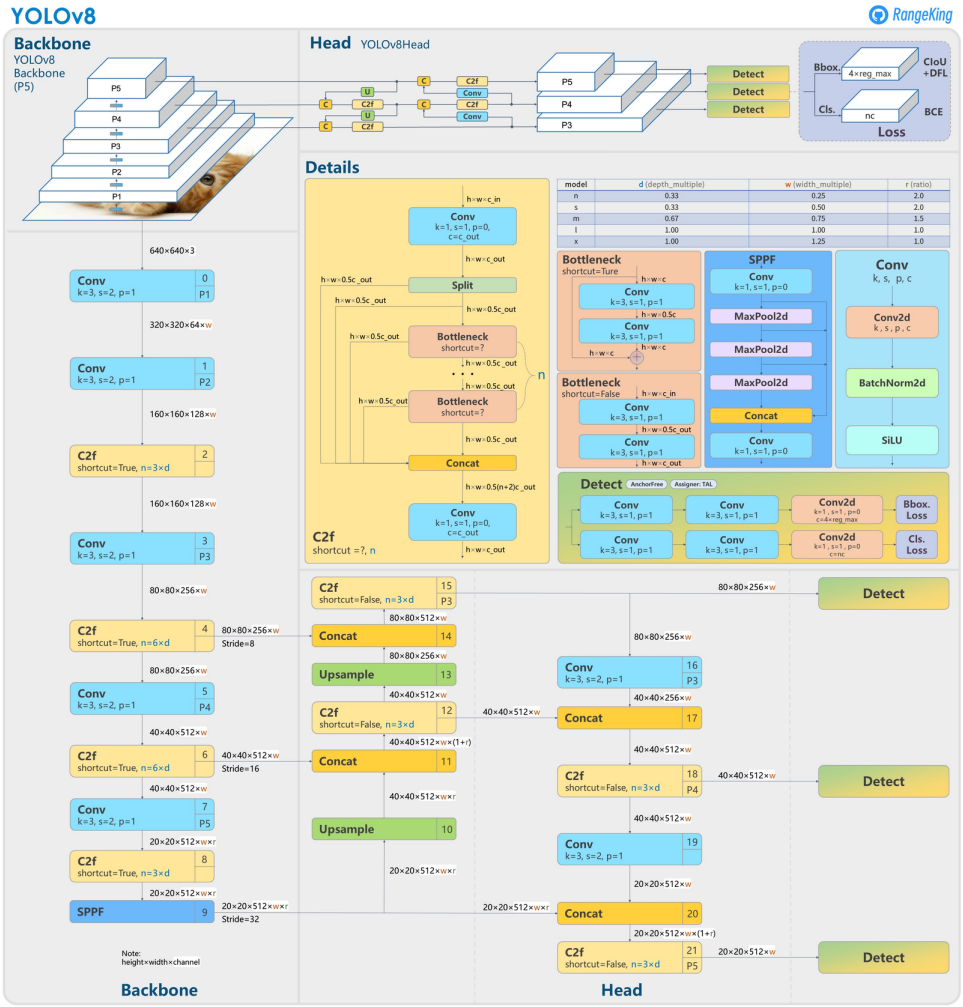


Figure A.1: **Detailed Architecture Diagram of YOLOv8**