SMALL OBJECT DETECTION ON UNCREWED SURFACE VESSEL

Final Report

THE UNIVERSITY OF SYDNEY

5703 Group Based Capstone Project

Group Members

1. Lewis Liu (500227956)
2. Kaiyuan Xu (500521681)
3. Zechen Zhu (540323234)
4. Jing Luo (540033724)
5. Taimur Enam (540479597)
6. Jiayuan Ma (520370197)

CONTRIBUTION STATEMENT

Our group CS5-2, taking project titled Small Object Detection on Uncrewed Surface Vessel, with group members Lewis Liu, Kaiyuan Xu, Zechen Zhu, Jing Luo, Taimur Enam, and Jiayuan Ma, would like to state the contributions that each group member has made for this project during this semester:

- Lewis Liu: GitHub Code Management, YOLO v8 v11 Model Training and Report Writing.
- Kaiyuan Xu:
- Jing Luo: Datasets Finding, MHT Model Training, Results Summarizing, and Report Writing.
- Jiayuan Ma: Monitoring Team Progress, Contacting the Client, Faster R-CNN Model Training, and Report Writing.
- Zechen Zhu:
- Taimur Enam: Data Sourcing & Analysis, CFAR Model Training and Report Writing.

All group members agreed on the contributions listed on this statement by each group member.

Signatures:

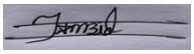
Jiayuan Ma (jima3429) – Jiayuan Ma

Zechen Zhu (zzhu0795) – zechen Zhu

Jing Luo (jluo0765) – JING LUO

Kaiyuan Xu (kaxu2143) – Kaiyuan Xu

Lewis Liu (rliu0639) – Runheng Liu

Taimur Enam (tena0927) – 

ABSTRACT

The abstract should be between 150-600 words. Briefly summarise your project/research. The abstract is usually written last, when you have a clear idea of your project as a whole. The aim of this section is to quickly introduce the reader to the project, and ideally engage their interest and encourage them to read the rest of the proposal. You should include an overview of the project, its motivation, the objectives, and the methods you have used, and discussions and findings. Do not include details in this section , you will have plenty of space in later sections. Also remember that the reader may not understand the technical details of your project so avoid jargon and leave in-depth discussion for later sections.

TABLE OF CONTENTS

1. INTRODUCTION

Uncrewed Surface Vessels (USVs) are the vessels which allow various modes in the platform without operators. Over the past few decades, USVs have been widely applied in the fields such as ocean environmental monitoring, rescue tasks and scientific surveys. As reviewed by Manley (2008), the system of USVs derived from autonomous surface craft (ASC) in the 1990s, which undertook the tasks like automated bathymetry experiments and acoustic transmissions undersea. The key technology for USVs is perception. It is vital for USV to possess environmental perception because it could achieve more effective autonomous operation and safe driving. The current research is still devoted to small object detection which could be utilized in USVs to avoid small obstacles.

However, in recent years, detecting small objects on water surfaces has become a major challenge based on vision due to factors such as light reflection, surrounding reflection and a short range of detection. An alternative approach to detecting small objects is to use sensor-based methods such as radar and camera. Radar could perform better and is more robust to lighting conditions than a vision-based system though it still has some difficulties to tackle such as the lack of semantic information. (Cheng et al., 2021).

Based on that, the project aims to concentrate on exploring radar-based small object detection using the MOANA dataset, a multi-modal maritime dataset incorporating both X-band and W-band radar, along with stereo cameras and LiDAR. Leveraging the rich and diverse data provided by MOANA, we experimented with a range of detection and tracking algorithms, including deep learning-based visual detectors such as YOLOv8, YOLOv11, EfficientDet, and Faster R-CNN, as well as radar-specific methods like CFAR and Multiple Hypothesis Tracking (MHT). Our approach involves adapting these models to radar input formats to achieve robust small object detection and tracking performance under complex maritime conditions such as multipath interference, clutter, and limited visibility.

2. RELATED LITERATURE

Small object detection in USV environments faces unique challenges due to factors like water reflections, low visibility, and dynamic scenes. To address these, researchers have developed two main types of approaches: vision-based models that

rely on camera input and deep learning, and sensor-based models that process data from LiDAR, radar, or IMU. This section reviews key works in both areas to understand how they contribute to improving detection accuracy and robustness.

2.1 Vision-Based Model

The early small objection tasks are dependent on traditional methods of machine learning before the rise of deep learning. Dalal and Triggs (2005) extracted handcrafted feature constructions from images by using histogram of oriented gradients (HOGs). Then the classifier like SVM could be used to recognize objects. This method could be applied in some simple scenes with specific backgrounds and clear target outlines with the advantage of its low computational cost (Zhang et al., 2023). However, due to the limitations of handcrafted features, such methods are insensitive to target scale variations and struggle to handle complex backgrounds and variable water surface reflections, which often leads to missed or false detections of small targets in real-world applications (Sun, H. et al., 2024).

Convolutional neural networks (CNNs) bring revolutionary improvements to visual object detection. The frameworks based on CNNs are separated into two categories: single-stage framework and two-stage framework. In particular, two stage framework algorithms like the series of R-CNNs are often used in the early deep learning visual models. The procedure is first generating region suggestions and then classifying them (Sun, X. et al., 2021). The typical method such as Faster R-CNN first extract candidate regions from the image and then extract and classify features of them. This method is suitable for medium and large objects. More complex approaches such as Cascade R-CNN and Mask R-CNN constructed feature pyramid structures (FPN) to adapt the small objects. However, it is hard to deploy in USVs due to the high requirements of hardware. They serve as a baseline for accuracy comparison across a wider range of scenarios (Sun, H. et al., 2024).

Compared with the two-stage framework, the single-stage framework such as the series of YOLO (You Only Look Once) and SSD, which can complete all the process of object detection at one stage (Lee et al., 2018). YOLO separates the images into a grid and directly predicts whether there's a target in each grid cell, along with its category and position. The early research attempted to apply YOLOv2 in the task of detection especially in the scenario where USVs require real-time detection because

the experiment shows that the algorithm achieves a good balance between accuracy and speed (Lee et al., 2018). More research surrounds YOLOv3, YOLOv4 and YOLOv5. Sun, X. et al. (2021) developed an improved YOLOv4-based model by incorporating a weighted cross-stage partial network (wCSPPAN) and a Feature Pyramid Transformer (FPT) to capture features of small objects and improve performance under adverse weather conditions by training the Singapore Maritime Dataset. Building on YOLOv5, Zhang et al. (2023) introduced Ghost modules and attention mechanisms and computed the anchor box again. They finally achieved a detection speed of 138 FPS. Another research based on YOLOv5 mentioned the loss function called WIOUv2 which could improve the robustness of the detection frame. YOLOv8 is a major evolution in the YOLO series, adopting a modular backbone–neck–head architecture to improve small object detection. Recent studies have proposed improved versions such as AB2D-YOLOv8, which integrates attention-based feature interaction (AIFI), dilation-wise residual modules (DWR), and a lightweight BiFPN fusion network. These enhancements significantly boost detection performance for dense and occluded maritime targets, achieving up to a 9% improvement in mAP on the SeaDroneSee dataset while maintaining real-time speed and low model complexity (Ming et al., 2024; Peng, Zhang, & Ma, 2024). YOLOv11 further advances the YOLO architecture by incorporating modules like Cross-Stage Partial Spatial Attention (C2PSA) and optimized loss functions (e.g., CIoU and DFL), enhancing its ability to detect small objects in cluttered and high-noise environments such as coastal SAR imagery. Experimental results show that YOLOv11 outperforms previous versions, achieving 90.5% mAP@50 and an F1-score of 0.86 on the HRSID dataset (Bakirci & Bayraktar, 2024; Ruan et al., 2025). Additionally, optimizer choice significantly affects YOLOv11's performance, with AdamW demonstrating superior accuracy and convergence, particularly in small object detection scenarios (Syamsul & Wibowo, 2024).

Another single-based framework model is EfficientDet. The model builds around an EfficientNet backbone. It also introduces BiFPN which is familiar with YOLOv8, and finally, with a compound scaling rule (Tan et al., 2020). However, the limitation is it is complex to implement, and it may still miss tiny objects (AlDahoul et al., 2022).

2.2 Sensor-Based Model

In the research of small objects detection, different types of sensors are widely applied to cope with complex marine environments. millimeter wave (MMW) radar, a type of radar, is well recognized for its adaptation of the illumination. The RISFNet model proposed by Cheng et al. (2021) transforms radar data into radar point cloud density map (RPDM) and fuses it with camera images, so that small floating objects on the water surface could be accurately detected under low visibility. LiDAR, also called Light Detection and Ranging, provides three-dimensional point cloud information. The Seal Pipeline proposed by Ahmed et al. (2024) combines LiDAR with inertial measurement (IMU) unit and Kalman filtering to improve the stability to track the objects in the condition that USVs keep operating. Additionally, Monocular Camera, as a type of camera, is also used to simulate depth perception. Bao et al. (2019) mentioned the model called MonoFENet. It predicts depth through RGB images and generates pseudo point clouds, which are then fused with semantic information to achieve 3D target detection.

In order to process these sensor data, researchers have proposed a variety of neural network structures optimized for small target detection. Point cloud processing networks such as PointNet++ are widely used in 3D detection tasks because they can retain the original point cloud geometry. On this basis, the PB-SOD model introduces a semantically guided sampling mechanism and a confidence perception module, which significantly improves the detection performance on sparse small targets (Dong & Li, 2024). RISFNet uses a multimodal fusion method based on the attention mechanism to fuse radar and image features, and introduces a time encoding module to enhance the temporal representation ability of the target (Cheng et al., 2021). Other structures such as DA-Net use a density-aware attention mechanism to alleviate the problem of missed detection of small targets caused by uneven distribution of point clouds (Wang et al., 2023). Geometric relationship network (GRNet) models the geometric dependency between points based on graph neural networks, combines local and global context features, and improves the geometric recognition ability of small targets (Li et al., 2020).

Classic radar algorithms are as follows. Multiple Hypothesis Tracking (MHT) is a radar-based method that generates and maintains multiple hypotheses. For each hypothesis, then applies the Hungarian matching algorithm to connect the data and

uses a Kalman filter to predict the states of objects. The strengths are Low cost to compute 4 and real-time. and the limitations are No appearance features, and Less robustness in dense targets (Zulkifley & Moran, 2012; Kim et al., 2015). The Constant False Alarm Rate (CFAR) algorithm is another widely used radar-based detection method, particularly in synthetic aperture radar (SAR) imagery for maritime surveillance. Traditional CFAR relies on fixed guard windows to estimate background clutter, which can lead to missed detections or false alarms when ship targets vary in size or are closely spaced. Dai et al. (2016) improved the CFAR detection process by combining it with object proposals. It treats each proposal region as a variable guard window, which makes it easier to detect ships of different sizes.

Overall, vision-based models offer efficient detection in clear conditions, while sensor-based methods enhance performance in more complex or low-visibility environments.

3. PROJECT PROBLEMS

3.1 Project Aims & Objectives

The primary goal of this project is to improve solar-powered uncrewed vessels of Ocius Technology to detect small, distant boats in ocean environments using 2D radar data. Current detection systems struggle with false alarms triggered by environmental noise like coastlines or rain, which can compromise surveillance efficiency. To resolve this, the project focuses on refining detection algorithms to distinguish real boats from noises while preserving the ability to spot objects at long ranges.

The objectives are as follows:

1. Data Collection and Preparation: Sourcing and analysing publicly available 2D maritime radar datasets such as MOANA to train object detection models. Additionally, providing additional data sources alongside the selected MOANA dataset.
2. Model Training: Training and testing several models with Deep Learning Techniques and Traditional Detection Algorithms, and analysing what works best for detection such as predicting object centres or bounding boxes.

3. Model Evaluation and Optimization: Evaluating object detection models performance by using metrics such as precision, recall, and mAP (mean average precision). Additionally, perform further fine tuning to reduce false alarms without missing actual detections.
4. Delivery: Providing a clear report on methodologies, model comparison results, limitations and future recommendation etc. Reusable code alongside datasets will be shared via GitHub as agreed.

3.2 Project Questions

The radar data currently in use by Ocius Technology is not a standard 3D point cloud. It is more comparable to 2D spatial coordinates. However, publicly available 2D maritime radar datasets with labels are comparatively small and it might affect the training of effective deep learning models. So it would be a great idea to develop at least one model using traditional algorithms and compare them with the deep learning models. In this project we explore whether deep learning models trained on a small dataset can outperform the models trained with traditional algorithms.

3.3 Project Scope

The scope involves sourcing and preprocessing publicly available radar datasets to train and evaluate different deep learning object detection models, which will be fine-tuned for a solar-powered uncrewed vessel called Bluebottle developed by Ocius Technology to detect maritime objects such as boats. Model performance will be evaluated using precision, recall, and mean average precision (mAP) metrics to balance accuracy and reliability. The final deliverables will include source code implementing the optimized algorithms alongside radar datasets and a scientific report that analyses the techniques explored and compares model results to provide actionable insights for improving radar detection systems.

4. METHODOLOGIES

4.1 Methods

4.1.1 YOLO

Yolo (You Only Look Once) is a single-stage object detection algorithm that can identify the category and bounding box of objects in an image with just one ‘look.’

Yolov8 is the latest Yolo series object detection algorithm launched by Ultralytics in 2023[2], which can be used for tasks such as image classification, object detection, and instance segmentation. In this project, I used yolov8 and the new yolo11, which I will introduce below.

YOLO v8 structure

Use YOLOv8 to automatically detect targets in the image, such as ships, buoys, kayaks, piers, and more. In this structure there is the classic three- stage structure of the YOL: Backbone + Neck + Head.

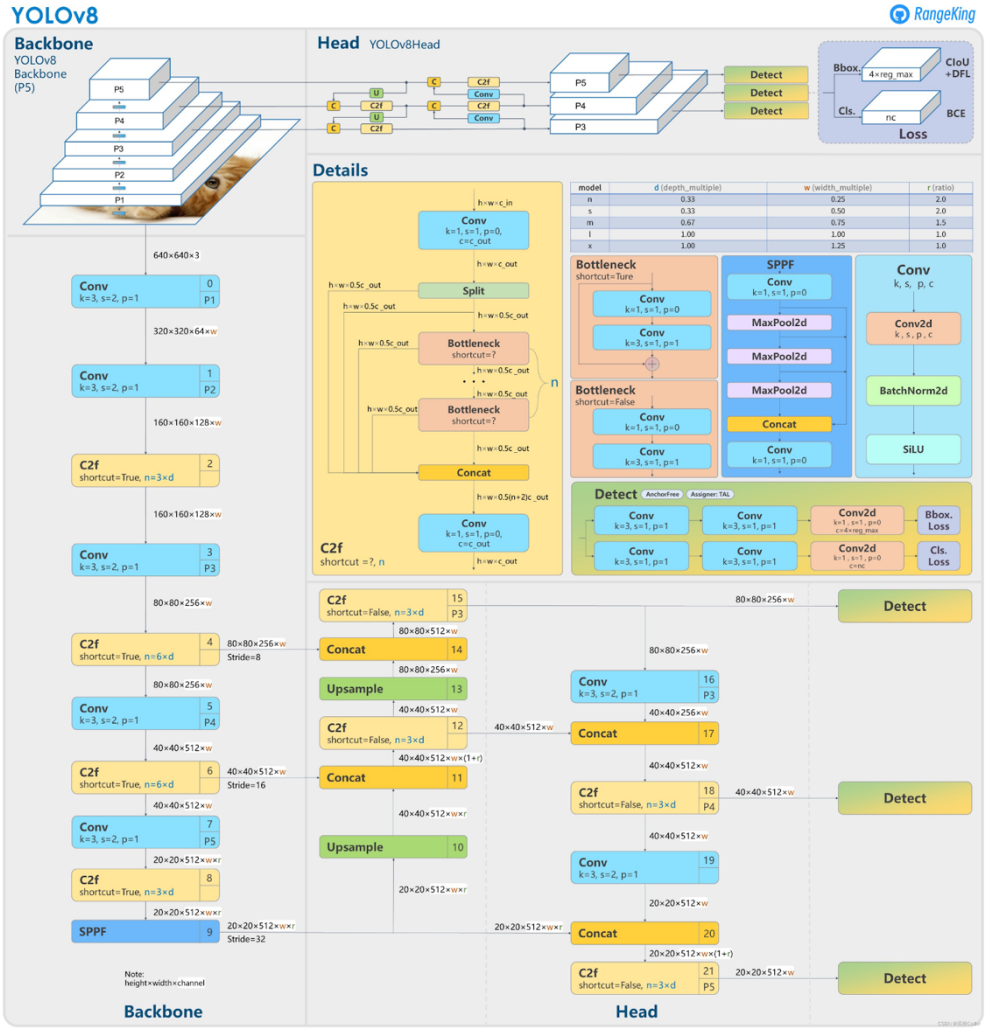


Figure 1: YOLO v8 Structure

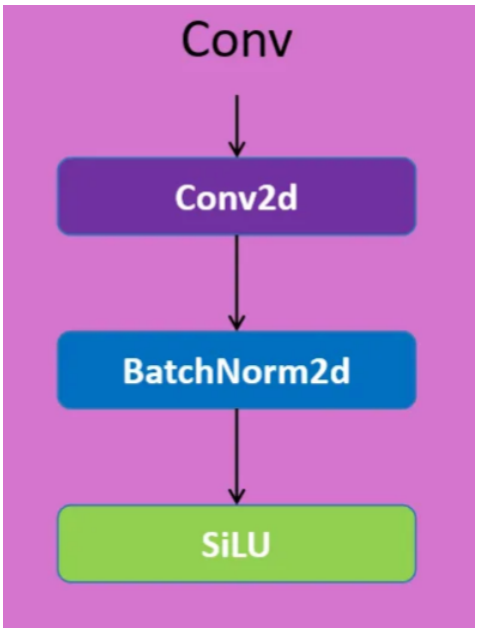


Figure 2: Convolutional

Convolutional Block: The activation function after each convolution is non-linear, enhancing the expressive power of the model. YOLO11 uses the SiLU (Sigmoid Linear Unit) activation function, which performs better than ReLU.

The activation ak of the kth SiLU for input Zk is computed by the sigmoid function multiplied by its input:

$$a_k(z_k) = z_k\sigma(z_k). \qquad \sigma(x) = \frac{1}{1+e^{-x}}.$$

Figure 3: Silu (left) and Sigmoid (right)

Backbone: The backbone is used to extract multi-level features from the input image, for example, Resnet or other neural networks are used to extract the global features, this piece is the main feature extraction, in yolov8 they use C2f. For example, enter an image of a water surface (e.g. 640x640 resolution) There may be multiple boats, buoys, or even piers or pedestrians in the image and Backbone will extract feature information from the image: such as edges, colour variations, shape and structure.

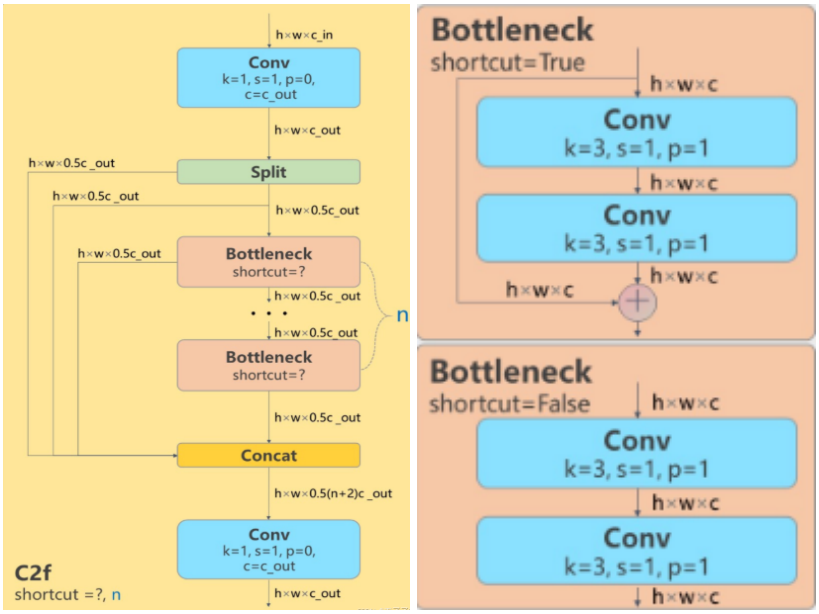
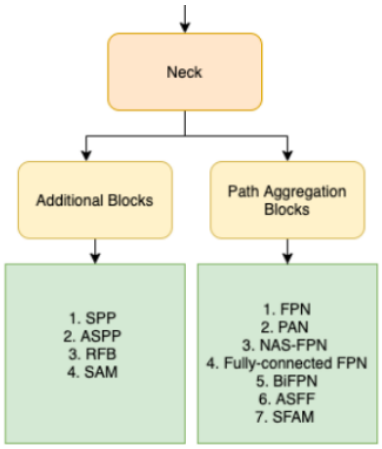


Figure 4: C2f and Bottleneck

C2f: As can be seen from the figure above, the number of channels in the input tensor of each Bottleneck in C2f is only half that of the previous level, resulting in a significant reduction in computational complexity. On the other hand, the increased gradient flow also significantly improves convergence speed and convergence effectiveness.

Bottleneck: Shortcut is a deep neural network structure design that directly adds input features to output features. It can solve the problem of gradient disappearance. When shortcut=True, the input is added to the output of the two convolutional layers to achieve residual connection, when shortcut=False, there are only two convolutional layers. Process part of them through a bottleneck layer while merging the other part with the bottleneck's output. This reduces computational load and improves feature representation.

Neck: After the features are extracted, some of them come from the image details, and Neck fuses these different levels of information, such as distance and size, so in Neck they include Additional Bocks, Path aggregation blocks.



$$F_i^{td} = \text{Upsample}(F_{i+1}^{td}) \oplus F_i$$

$$F_i^{bu} = \text{Downsample}(F_{i-1}^{bu}) \oplus F_i^{td}$$

Figure 5: Neck

Head: Head is the output module of YOLOv8 that makes judgments about each location: e.g. where there is a target, what category it is, and where the location is.

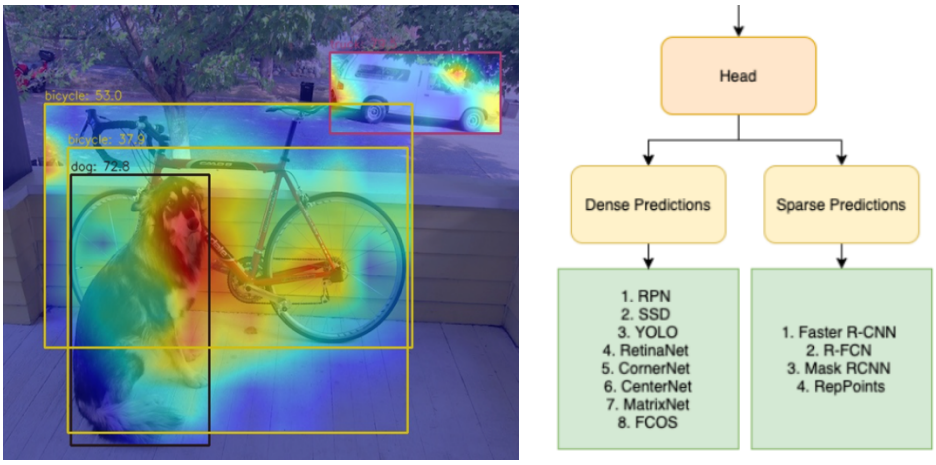


Figure 6: Head

YOLO Three Loss Functions: YOLO's loss functions typically consist of three components: classification loss (cls_loss), bounding box loss (box_loss), and distribution focal loss (dfl_loss).

$$\text{Box Loss} = \lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} 1_{ij}^{\text{obj}} \left((x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 \right) + \lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} 1_{ij}^{\text{obj}} \left((\sqrt{w_i} - \sqrt{\hat{w}_i})^2 + (\sqrt{h_i} - \sqrt{\hat{h}_i})^2 \right)$$

Figure 7: box_loss

box_loss (Bounding Box Regression Loss): Used to optimise the difference between predicted bounding boxes and ground truth bounding boxes.

$$\text{Classification Loss} = \sum_{i=0}^{S^2} 1_i^{\text{obj}} \sum_{c \in \text{classes}} (p_i(c) - \hat{p}_i(c))^2$$

Figure 8: cls_loss

cls_loss (Classification Loss): Optimises the model's prediction accuracy for object categories, ensuring the model can correctly identify the object categories in an image.

$$\text{DFL} = -\sum_{i=1}^{N} \sum_{c=1}^{C} y_{ic} \left(\alpha(1 - p_{ic})^{\gamma} \log(p_{ic}) + (1 - \alpha) p_{ic}^{\gamma} \log(1 - p_{ic})\right)$$

Figure 9: dfl_loss

dfl_loss (Distribution Focal Loss): A special loss function in the YOLO series, primarily used to address the class imbalance issue in object detection and improve the model's performance when handling difficult samples.

YOLO11 structure

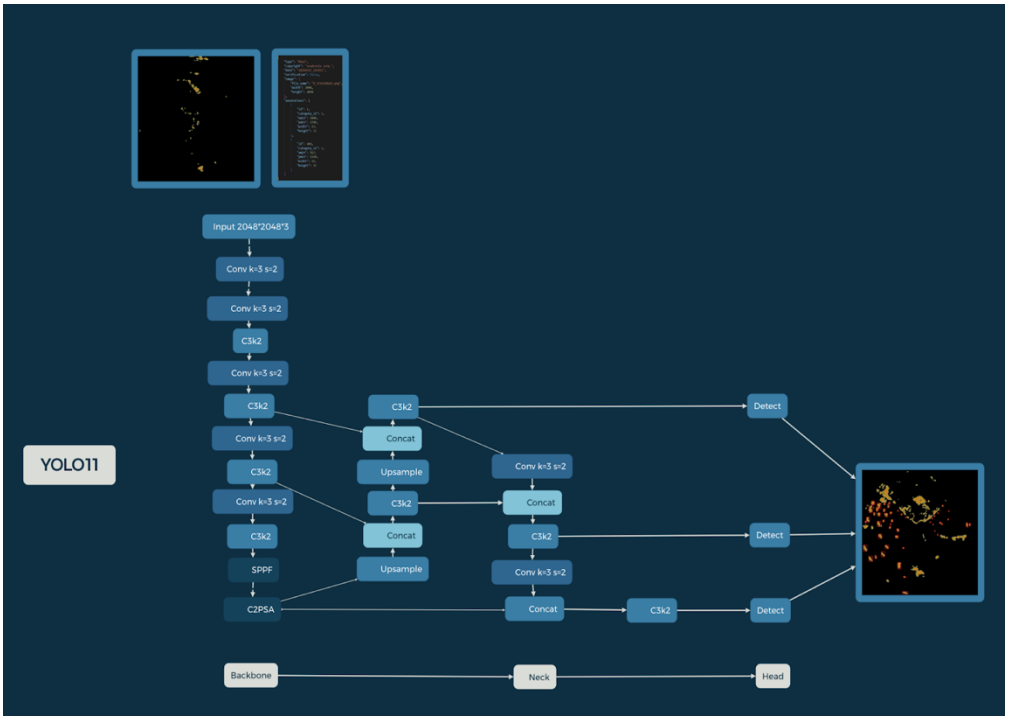


Figure 10: YOLO v11 Structure

YOLO v11 still uses the three-part backbone-neck-head structure of the YOLO series, but there are some minor changes to the backbone and neck, which I will introduce next.

Backbone: As we can see from the structure diagram, the difference between YOLOv8 and C3k2 lies in the use of C2f neural layers in YOLOv8. During the SPFF process, the C2PSA attention mechanism is applied to enhance feature expression capabilities. I will now explain these two aspects in detail.

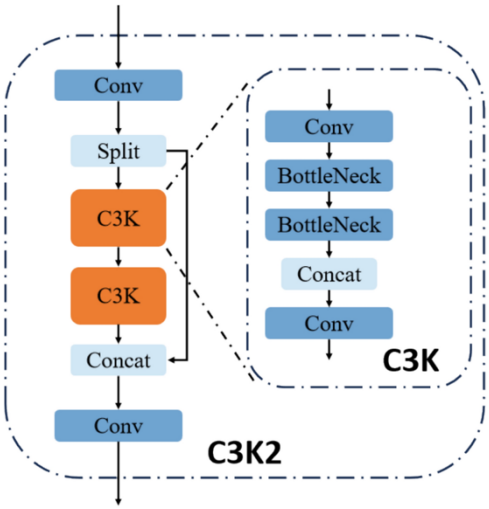


Figure 11: C3K2

C3k2: YOLO11 uses C3k2 blocks instead of C2f, which is more computationally efficient. This block is a custom implementation of the CSP Bottleneck, using two convolutions instead of a single large convolution (as in YOLOv8).

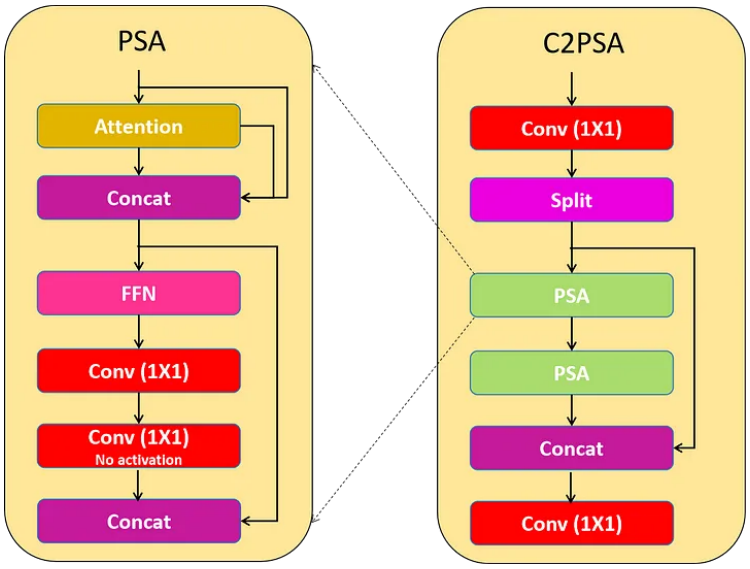


Figure 12: C2PSA

C2PSA: The C2PSA (cross-stage local spatial attention) module enhances spatial attention in feature maps, improving the model's focus on important parts of

images. Through spatial pooling features, the model can more effectively focus on specific areas of interest.

Neck: The neck is responsible for aggregating features of different resolutions and passing them to the head for prediction. It typically involves upsampling and connecting feature maps of different levels. It is basically the same as YOLOv8, except that the neural layer has been changed from C2F to C3K2.

4.1.2 Faster R-CNN

Faster R-CNN is a refinement of Fast R-CNN. The whole process is shown in the figure below. The input image is first passed through a backbone convolutional network (ResNet-50 by default) to extract multi-level feature maps, and then this shared feature map serves two tasks simultaneously. One is a Region Proposal Network (RPN) that sits on top of the feature map. RPN projects k = 9 anchor boxes at the center of each 3 × 3 sliding window, quickly determines whether these anchor boxes contain the foreground or not, and gives a rough geometric offset, resulting in around 300 high-quality proposals. Secondly, on the basis of proposals, regions of interest of any size are uniformly trimmed into a fixed 7 × 7 tensor by RoI Pooling, and then sent to the fully connected layer for fine classification and quadratic bounding box regression. The bottom-up yellow arrows emphasize feature sharing: the RPN and the detection head rely on the same convolutional feature map, thus eliminating double computation and preserving the possibility of joint end-to-end optimization.

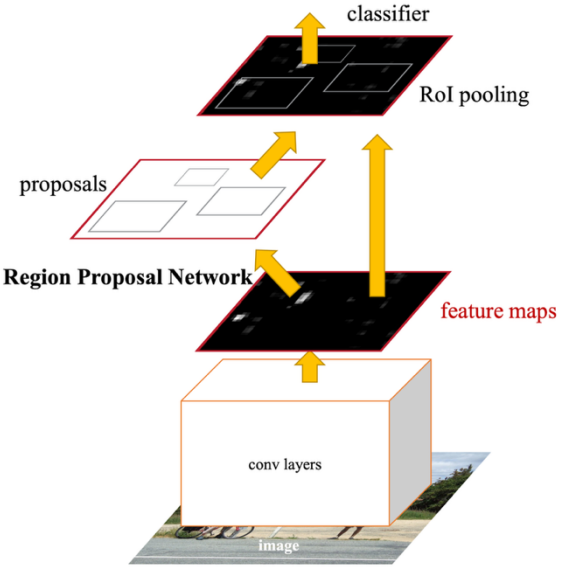


Figure 13: Faster R-CNN is a single, unified network for object detection. The RPN module serves as the 'attention' of this unified network.

In the figure below, for RPN, the feature map first goes through a 256-dimensional intermediate convolution layer at the sliding window, and then bifurcates into two parallel 1 × 1 convolution heads. The classification head outputs 2k foreground/background scores, while the regression head outputs 4k coordinate increments, each corresponding to k anchor boxes in the center of the sliding window. The multi-scale and multi-aspect ratio anchor box illustrated by the blue dotted line makes the network naturally have the initial coverage ability for different object sizes and shapes. The figure below also shows an object detection example based on these RPN proposals. Faster R-CNN can accurately locate the targets with wide scales in the same image, which verifies the effectiveness of the two-level regression strategy of "coarse adjustment + fine adjustment".

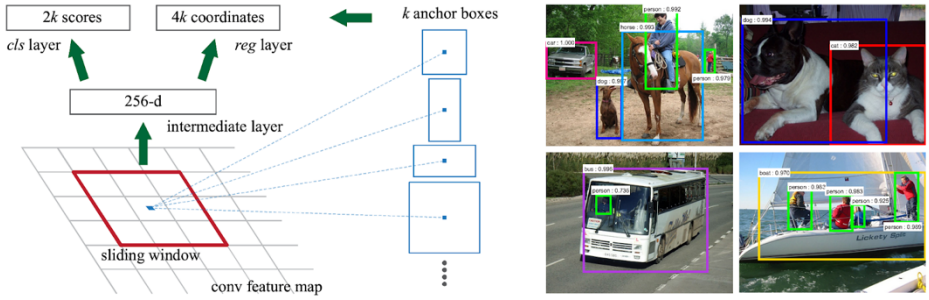


Figure 14: The left figure shows Region Proposal Network (RPN). The right figure shows example detections using RPN proposals on PASCALVOC 2007 test. Faster R-CNN detects objects in a wide range of scales and aspect ratios.

The innovation of Faster R-CNN is to combine the "Find locations of possible objects" (RPN) and the "judge and refine these locations" (classification/regression head) into the same network, and both share convolutional features and are jointly trained with a set of multi-task losses. In this way, it not only inherits the high accuracy of the two-stage detection, but also significantly reduces the overhead of candidate region generation, achieving the fastest and most accurate general object detection at that time.

4.1.3 EfficientDet

The design basis of EfficientDet is derived from the model EfficientNet in image classification. EfficientNet is a lightweight and efficient image classification network

proposed by the Google Brain team. It is generated by neural architecture search (NAS) and uses a unified scaling strategy called Compound Scaling. EfficientDet is an efficient and scalable target detection framework proposed by the same team (Google Brain) in 2020. It inherits EfficientNet as its backbone network and applies its feature extraction capabilities to target detection tasks, providing a solid feature foundation for multi-scale feature fusion and target prediction.

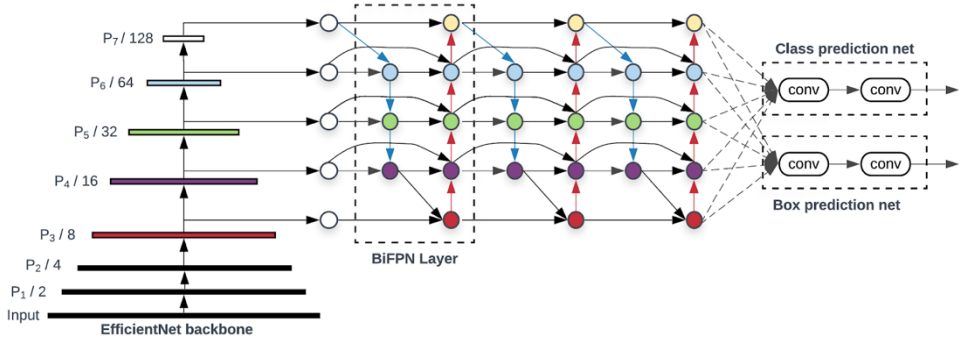


Figure 15: EfficientDet Architecture

In the EfficientDet architecture, as shown in Figure 3 above, the EfficientNet backbone network first processes the input image, and feature maps of different resolutions from P3 to P7 are output, corresponding to different levels of semantic information in the backbone network. Subsequently, these feature maps are sent to the BiFPN (Bidirectional Feature Pyramid Network) module for fusion. Moreover, BiFPN is one of the core innovations of EfficientDet. It not only supports bidirectional information flow from top to bottom and bottom to top but also introduces learnable fusion weights, which enables the network to adaptively weight fusion according to the importance of different feature layers. Compared with the traditional feature pyramid network, BiFPN has a more regular structure, more streamlined connections, and uses depthwise separable convolution to further reduce the amount of computation. These BiFPN layers can be stacked multiple times to enhance multi-scale object recognition capabilities.

Additionally, Figure 4 above compares the BiFPN used by EfficientDet with other mainstream feature fusion network structures, including traditional FPN (a), PANet (b), and NAS-FPN (c). The FPN structure only has a top-down information flow, and the information fusion is incomplete; PANet introduces an additional bottom-up path, which is richer in information but more computationally expensive; although NAS-FPN has a highly optimized structure, it relies on the automatic search

of a large number of computing resources. In comparison, BiFPN (Figure d) achieves the unity of efficiency and expressiveness in structural design. Not only does it have a clear fusion path and can be stacked repeatedly, but it also improves the flexibility and interpretability of information fusion through the weight mechanism.

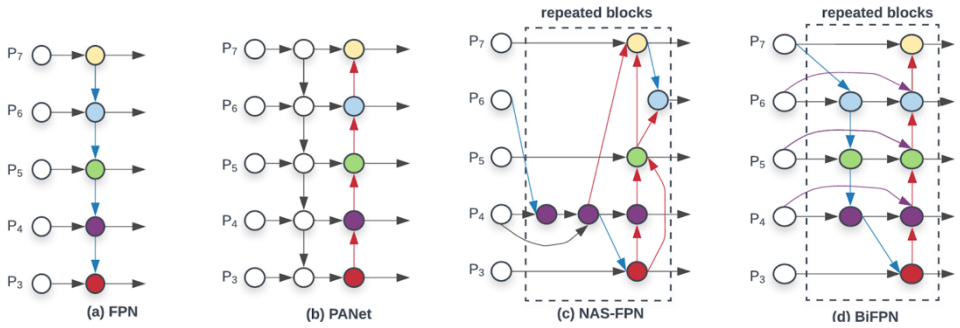


Figure 16: Feature Network Design

According to these, EfficientDet is not a simple splicing based on existing models but combines the efficient backbone capabilities of EfficientNet with the newly designed BiFPN module, and achieves global optimization of the model through a unified scaling strategy, thus building a set of target detection models with excellent performance.

4.1.4 MTH

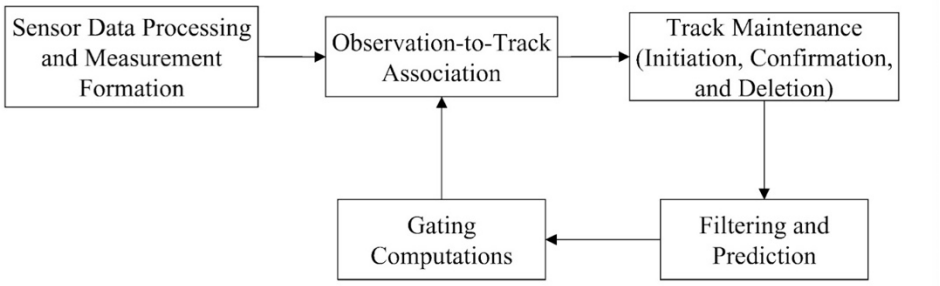


Figure 17: Basic Elements of a Conventional MTT System

MHT, or multiple hypothesis tracking, is a well-established and reliable multi-target tracking (MTT) technique. Global hypothesis management and postponed decision making are its fundamental concepts. MHT retains multiple possible paths of data association in each frame, progressively grows to form multiple global trajectory hypothesis trees, and then uniformly evaluates and prunes them to effectively deal with complex situations like occlusion, target intersection, or detection error. This is in

contrast to traditional methods like Hungarian or GNN, which rely on greedy decisions in the current frame.

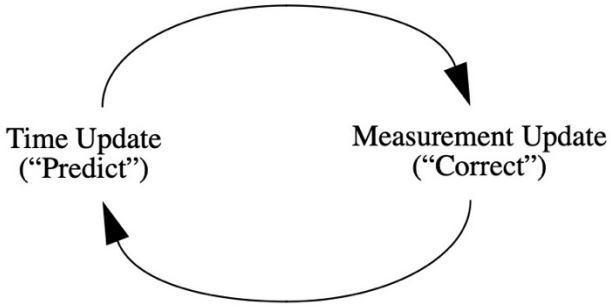


Figure 18: The Ongoing Discrete Kalman Filter Cycle

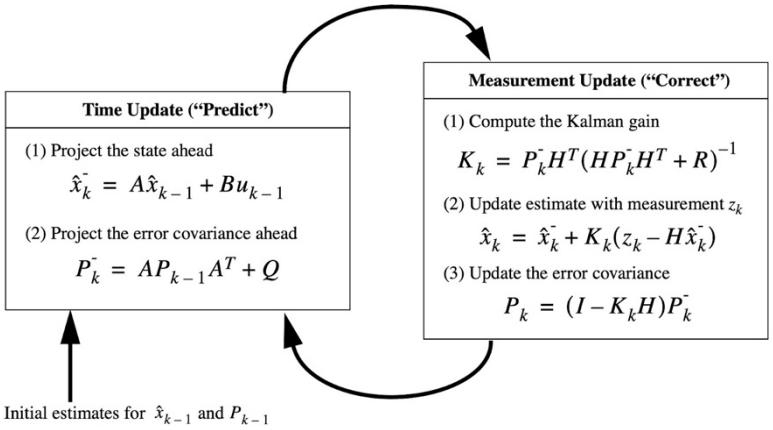


Figure 19: A Complete Picture of the Operation of the Kalman Filter

For this project, we used an integrated Kalman filter and an MHT tracker based on the IOU distance metric to represent the target's motion state. We also presented the Murty algorithm to broaden trajectory hypotheses, acquire many suboptimal solutions, and execute Top-K optimal matching search on the association cost matrix, therefore increasing the flexibility and robustness of multi-target data association.

To keep all global hypothesis trees, we constructed a RealMHT class as part of the overall architecture design. The observation box of the current frame, the target ID, the parent node pointer, the Kalman filter state, and a scoring indicator are all included in the HypothesisNode representation of each hypothesis node. State prediction, data association, multiple match generation, hypothesis expansion, new target initialization,

and hypothesis trimming are all sequentially carried out by the tracker whenever a new frame is received.

Initially, state prediction is carried out, and the projected box is obtained by extrapolating each target's position in the historical trajectory using the Kalman Filter. Each target's predicted status in the current frame is provided in this phase.

The data association comes next. The cost matrix between each forecast box and the current frame's detection box is determined using the IOU distance. Unlike the traditional Hungarian approach, we use the Murty algorithm to generate numerous alternative matching combinations rather than relying on a single ideal match. The basic idea behind the Murty algorithm is to start with the best solution that the Hungarian algorithm produced, then gradually alter one of the matching relationships, create a new alternative solution, and sort it by total cost until the Top-K optimal or suboptimal solutions are found. The hypothesis tree is grown by creating new child nodes for every matching combination.

The detection boxes that do not match are regarded as possible new targets, and new trajectory branches are initialized for them. If no detection matches the current trajectory, we will also keep the hypothesis and let it be successfully re-matched in subsequent frames.

To manage the computational complexity, the system incorporates a scoring mechanism to assess each global hypothesis path's believability. Indicators like the matching IOU, Kalman filter confidence, or target survival time may be used to determine the score. Only the K global hypothesis paths with the greatest scores at the end of each frame are kept for the subsequent expansion cycle.

One of the most important parts of the whole system is the Kalman Filter. We retain a 7-dimensional state vector in each HypothesisNode that includes the target center location (x, y), area s, aspect ratio r, and its first-order derivative, which is utilized to describe the target's smooth movement. In order to improve the matching accuracy and trajectory continuity, the detection box is sent into the filter via the measurement vector (x, y, s, r). This updates the target state.

The system will determine the trajectory path of each target by tracing back each hypothesis tree from the root node to the leaf node after processing all the frames. In order to guarantee that the trajectory is continuous, stable, and as comprehensive as

feasible, we choose the branch with the longest path length as the final output for the same target ID if there are several.

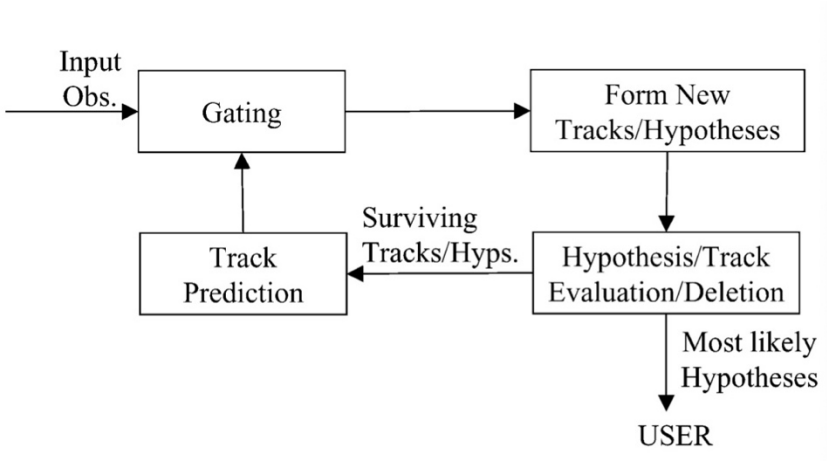


Figure 20: MHT Logic Overview

Generally speaking, MHT offers a more robust and worldwide multi-target tracking system. It exhibits notable benefits in the face of issues like false positives and short-term occlusion because it holds onto several options until sufficient data is gathered rather than making a decision in a rush in each frame.

4.1.5 CA-CFAR

Cell-Averaging Constant False Alarm Rate (CA-CFAR) is a variant of CFAR - a traditional radar target detection method. The purpose of CA-CFAR is to keep the probability of false alarm (Pfa) fixed in changing noise and clutter conditions. To achieve this, it adaptively sets the detection threshold based on local noise estimation.

CFAR Window Configuration

CFAR scans radar data with a sliding window which includes CUT, guard cells and training cells.

Cell Under Test (CUT): The particular data point to be tested as a possible target.

Guard Cells: They are the adjacent cells to the CUT. They're not included in the estimation of noise to ensure the target signal does not bias the result.

Training Cells: These are found around the guard cells and are utilized to estimate the background level of the noise.

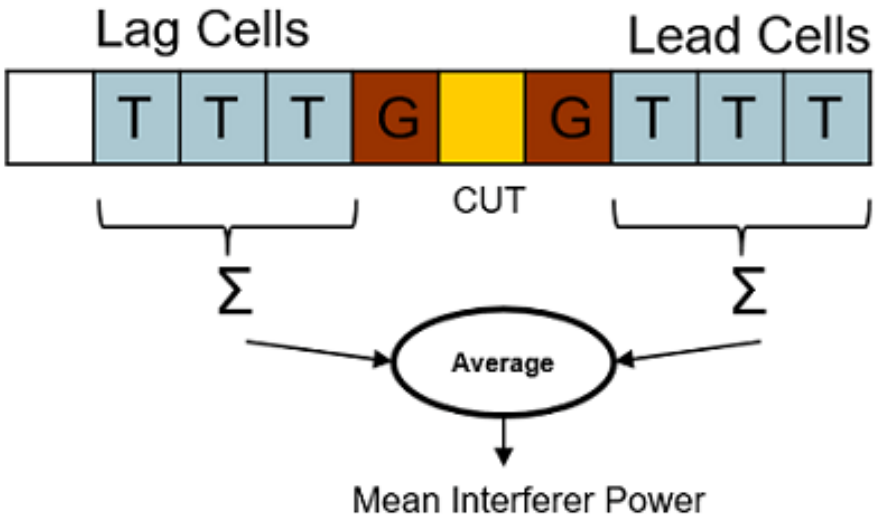


Figure 21: Cells

Background Noise

The average background noise around the centre pixel is calculated using $\widehat{P_n^2} = \frac{1}{N}\sum_{i=1}^{N} x_i$, where Pn is the estimated noise power, N is the number of training cells, and xi represents the signal strength in the training cell number i.

Calculate Threshold

The detection threshold is calculated by scaling the estimated noise power. It automatically generates a detection threshold for each pixel based on the calculated noise.

Detection Decision

The signal power within the CUT is compared to the threshold:

xCUT > T, then a target is detected.

xCUT $\leqslant$ T, the target is not detected.

4.2 Data Collection

In this project, we research and study multiple publicly available maritime radar datasets for object detection. We shortlist and analyse three datasets including MOANA, SAR-Ship-Dataset and Dual-polarimetric SAR Dataset before selecting the best one for model training.

MOANA Dataset

The MOANA (Multi-Radar Dataset for Maritime Odometry and Autonomous Navigation Application) dataset has been sourced from a research paper at MOANA from arXiv, and the dataset is available at MOANA Dataset.

MOANA dataset was collected across seven voyages near South Korea and Singapore by capturing multimodal sensor data using two vessels. Each frame contains four types of imagery: X- band radar, W-band radar, binocular camera, and short-range LiDAR, along with GNSS reference trajectories and external calibration files. Among them, this section of navigation data from the X-band radar of a single island contains the most detailed label for 1655 images.

SAR-Ship-Dataset

The SAR-Ship-Dataset has been sourced from a GitHub repository which is available at CAESAR-Radi. The dataset is created by using 102 Chinese Gaofen-3 images and 108 Sentinel-1 images. The dataset contains 39,729 images and the labels are in txt format and contain the category of the ship, normalized column and row values of the ship center and normalized height and width of the ship.

Dual Polarimetric SAR

The Dual Polarimetric SAR dataset has been sourced through a research paper at MDPI and the dataset is available at liyiniiecas. The dataset has 1236 images (256x256 pixels) and all the images are cropped from 50 dual-polarimetric SAR images from Sentinel-1. The labels are in XML format and provides both RBox (center coordinates, height, width, and angle of a box) and BBox (top left corner and the lower right corner coordinates of a box)

4.3 Data Analysis & Visualizations

We analyse the data using Python by generating statistical summary outputs, bar plots, time series plots, heatmaps, and scatter plots to get meaningful insights.

MOANA Dataset

X-band radar of a single island in the MOANA dataset contains the most detailed labels. Data labels are in JSON format. This section of the dataset has 1655 images with labels.

Figure 22: A Raw Image with Two Objects (Annotations) and Its Label

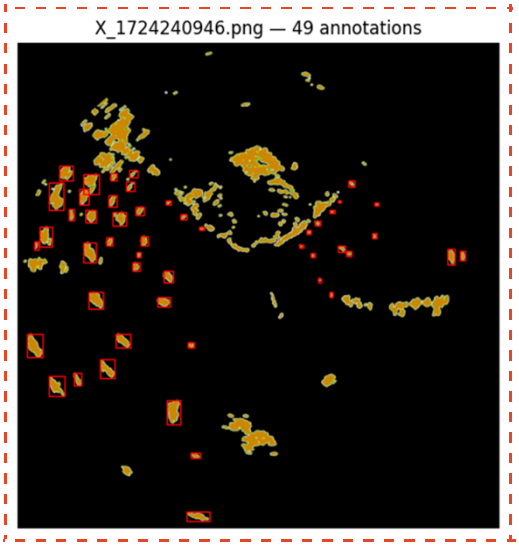


Figure 23: An Image with Identified Bounding Boxes (49 Ships and Buoys in This Image)

The number of annotations/ bounding boxes per image are between 0 and 49. There are 144 images with an empty annotation list (no vessels, buoys), 104 (6.28% of the total data) images with just one annotation (1 object), and 7 images with 49 annotations.

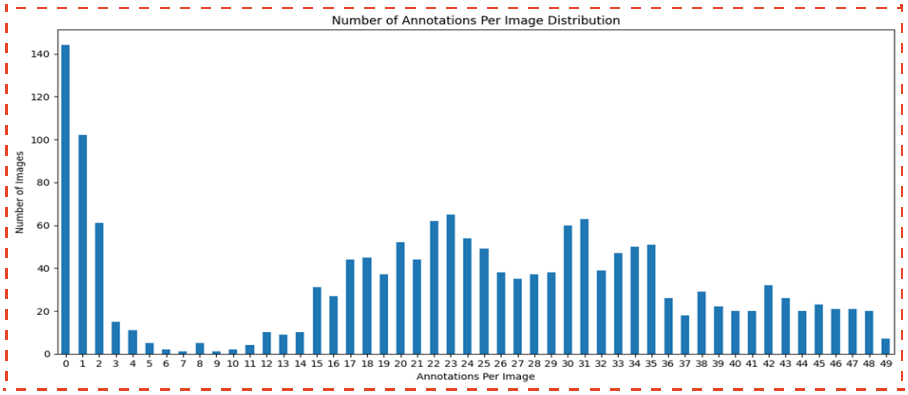


Figure 24: Annotations Range Distribution

There is only one ‘category_id’ and it's labeled as 1 which means object, and 105 unique ‘id’ which indicates 105 unique objects (ships and buoys). There are in total 38,272 ship instances (bounding boxes). Additionally, no JSON file contains duplicate ‘id’ values which means there is no data duplication.

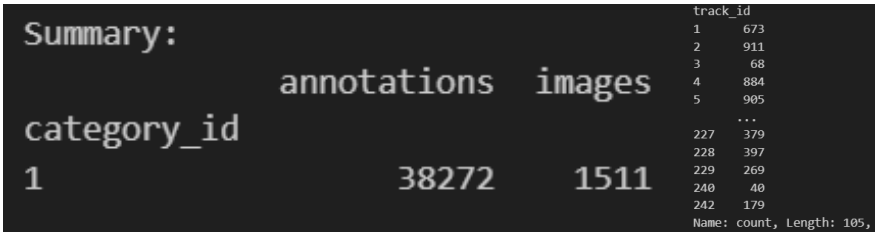


Figure 25: Summary Output to Count category_id and object id

The below scatter plot indicates the wider boxes tend to be taller which might be large vessels and there’s a long tail near zero (not 0 since it’s been excluded from this plot) which might be small buoys.

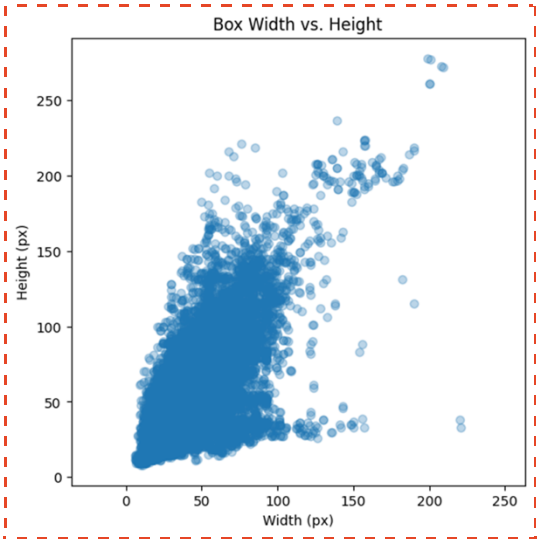


Figure 26: Box Width v.s. Height

The image frames were taken between 14:39:11 and 14:45:31 on 27 March 2025 and there are 352 unique times of the day. It might indicate that the images are taken from a 6 minutes long video.

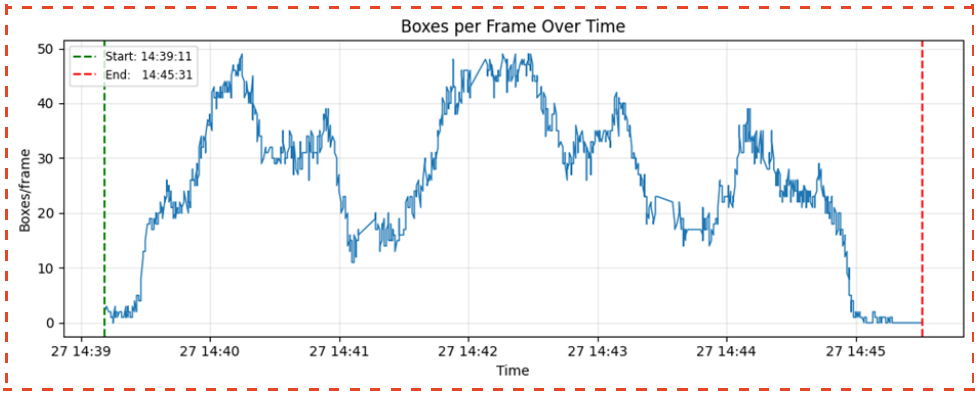
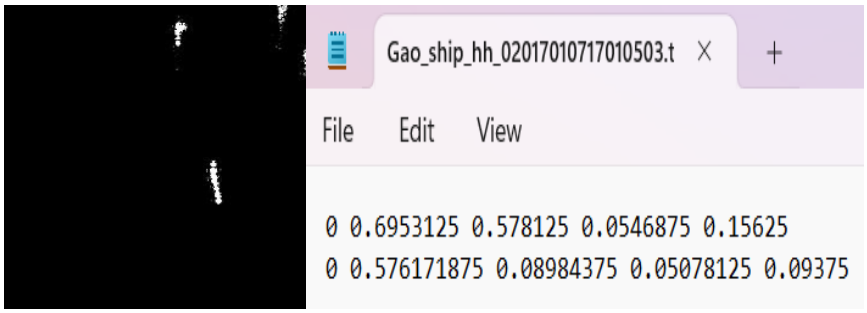


Figure 27: Boxes Per Frame Over Time

SAR-Ship-Dataset

The SAR-Ship-Dataset contains 39,729 images with labels and the labels are in txt format.



Figures 28: A Raw Image with Two Objects (Annotations) and Its Label

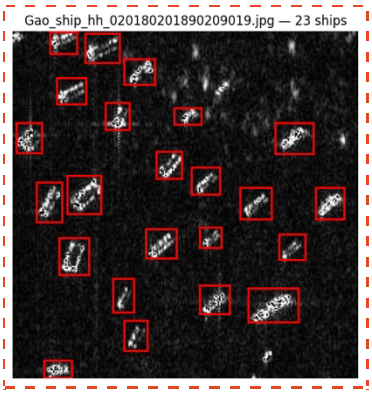


Figure 29: An Image with Identified Bounding Boxes (23 Ships in This Image)

The number of annotations/ bounding boxes per image are between 1 and 23 but the distribution is not balanced. There are 33,627 (84.64% of the total data, which is

significantly high) images with one ship annotation and 4 images with 23 ship annotations. There are in total 50,885 ship instances (bounding boxes).

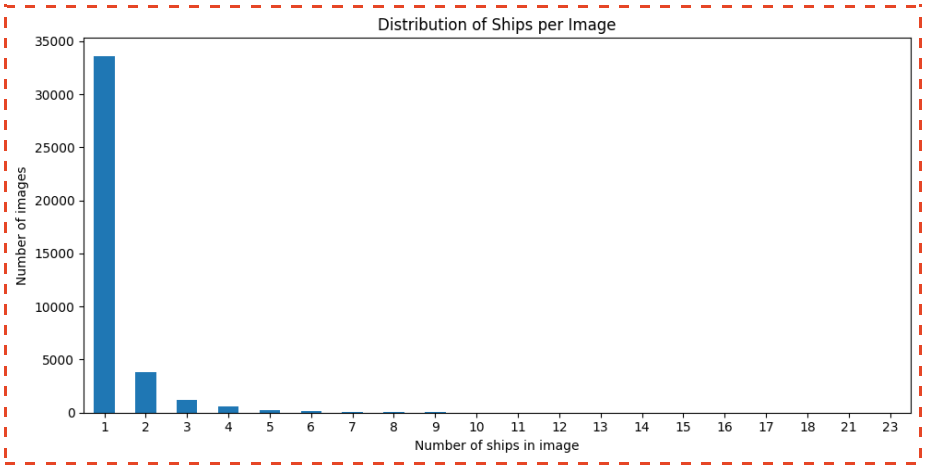


Figure 30: Annotations Range Distribution

50885 total ship annotations over 39729 images

	image	class	x_center	y_center	width	height
0	Gao_ship_hh_020160825440101020.jpg	0	0.923828	0.845703	0.121094	0.128906
1	Gao_ship_hh_020160825440101021.jpg	0	0.419922	0.845703	0.128906	0.128906
2	Gao_ship_hh_020160825440101037.jpg	0	0.582031	0.542969	0.117188	0.078125
3	Gao_ship_hh_020160825440101038.jpg	0	0.082031	0.542969	0.117188	0.078125
4	Gao_ship_hh_020160825440101039.jpg	0	0.562500	0.886719	0.101562	0.078125

Figure 31: Dataframe Output

Dual-polarimetric SAR Dataset

Dual-polarimetric SAR Ship Detection Dataset contains 1236 images and the dataset is already splitted into train and test sets where the train set contains 856 images and the test set contains 380 images. The labels are in XML format and the dataset provides two types of labels for both train set and test set. Ships are labeled with both a rotatable bounding box (RBox) and a horizontal bounding box (BBox).

Figure 32: A Raw Image with 1 Ship from the Train Set with both RBox and BBox Labels

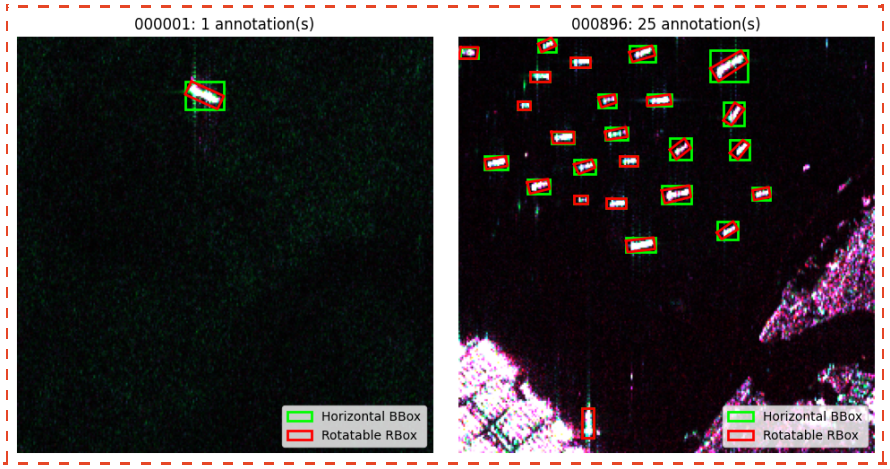


Figure 33: Two Images with Identified Bounding Boxes (1 Ship & 25 Ships Where Red Boxes Are RBox and the Green Boxes Are BBox)

The number of annotations/ bounding boxes per image are between 1 and 25. There are 542 (43.85% of the total data) images with just one ship annotation and just one image with max annotations (25 ships).

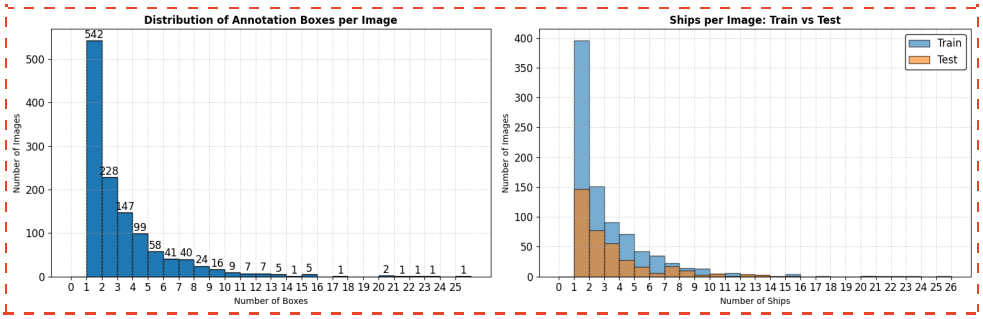


Figure 34: Annotations Range Distribution (Overall and Train v.s. Test)

The dataset includes 3540 ship instances in total with 2337 annotation boxes in the train set and 1203 annotation boxes in the test set. There is only one unique <name> value which is 'ship' and there is data duplication.

Horizontal BBoxes:

	image	xmin	ymin	xmax	ymax	width(xmax-xmin)	height(ymax-ymin)
0	000001	103.0	27.0	127.0	44.0	24.0	17.0
1	000002	59.0	97.0	77.0	115.0	18.0	18.0
2	000003	175.0	157.0	193.0	174.0	18.0	17.0
3	000004	77.0	116.0	90.0	136.0	13.0	20.0
4	000005	44.0	24.0	64.0	43.0	20.0	19.0

Rotatable RBoxes:

	image	cx	cy	width	height	angle_rad	angle_deg
0	000001	115.5626	36.0569	8.7642	22.4718	2.006508	114.964440
1	000002	67.9324	105.9189	9.1208	20.4160	0.970797	55.622552
2	000003	184.5000	165.8830	7.5305	19.8064	0.809214	46.364547
3	000004	84.6403	126.4675	8.5277	19.3560	0.413217	23.675590
4	000005	54.3456	33.3283	8.6905	20.2676	0.891416	51.074375

Figure 35: Dataframe Output for both BBox and RBox

From the analysis and visualisations of three shortlisted datasets a comparison table has been provided below:

Comparative Analysis	MOANA	SAR Ship Dataset	Dual-Polarimetric SAR
Total images with labels	1655	39,729	1236
Annotations format	JSON	TXT	XML
Annotations per image range	0 to 49	1 to 23	1 to 25
Num of images without objects	144	0	0
Num of images with just one object	104 (6.28%)	33,627 (84.64%)	542 (43.85%)
Num of images with max objects	7 (49 annotations)	4 (23 annotations)	1 (25 annotations)

Total objects	38,272 ships and buoys	50,885 ships	3540 ships
Unique class id (Object)	1 as '1'	1 as '0'	1 as 'ship'
Unique id for different objects	105 unique ships/ buoys	NA	NA
Unique Time & Date of the day	352	NA	NA
Data duplication	No	No	No

After careful consideration, we choose X-band radar imagery from the MOANA dataset to train the models because:

1. It satisfies 2D radar maritime data requirements and is similar to the data client using.
2. The annotations distribution range in images are balanced compared to other two datasets, such as in SAR-Ship-Dataset 84.64% of the total images contains only 1 ship annotation which is not a balanced distribution.
3. Includes some images without objects (empty annotation list) which helps the models learn what not to detect.
4. It provides two types of labelled objects such as Ship & Buoys.
5. Includes detailed object labels

4.4 Deployment

Explain the deployment of the system on client's infrastructure. Remember to mention how updates and bug fixes will be distributed after the deployment. Attention please: some of our projects may not involve deployment process. If that is the case, your group is allowed to provide brief reason on why deployment process is not necessary in your project as the content for this subsection.

4.5 Testing

Include the detailed description of your testing process and methodologies. Examples include: Test Driven Development (TDD), Unit Testing, Integration Testing, vigorous testing, etc. Remember to explain why the methodology was chosen and how the defined testing process will contribute to the quality software development in your specific project. Attention please: some of our projects may not involve testing process. If that is the case, your group is allowed to provide brief reason on why testing process is not necessary in your project as the content for this subsection.

5. RESOURCES

Indicate the resources that will be required to complete your projects. Depending on the nature of the project, include sections where appropriate.

5.1 Hardware & Software

Hardware: The client provides a gaming PC with a 16GB NVIDIA GeForce RTX 4060Ti, which is sufficient for most medium-sized deep learning tasks. The team members also had their own personal computers, including the gaming PC and MacBook Pro with Apple Chips, which allowed for small-scale experiments, preliminary model exploration, and simplified models with low resource requirements.

Software: Our software workflows are modelled and trained around PyCharm and Visual Studio Code, both of which provide powerful debugging, library integration, and version control support. We rely on GitHub to manage our codebase, support branches, pull requests, and efficient collaboration among team members. In addition, we use Google Drive to share documents, datasets, and other project-related files, ensuring seamless communication and organisation during development.

5.2 Materials

In addition to hardware and software, the following resources are also required:

1. Computational Resources

- High-performance GPU (e.g., NVIDIA RTX 4060Ti, A100, or cloud-based solutions like AWS/GCP Colab Pro) is used to efficiently train deep learning models, especially when processing 3D point cloud data, which requires a lot of computation.
- Cloud Computing Services (e.g., AWS EC2, Google Cloud, Microsoft Azure, or university-provided HPC clusters) If local computing power is limited, cloud computing resources can be used for training and testing.

2. Data and Storage

- High-Capacity Storage (HDD/SSD or Cloud Storage like Google Drive, OneDrive, or AWS S3)
- Store large-scale data sets and checkpoints of training models.

3. Development and Collaboration Tools

- Jupyter Notebook / VS Code / PyCharm: Code development and debugging environment.
- GitHub / GitLab / Bitbucket: Code management, version control and team collaboration.
- Google Docs / MS Word: Write project reports, papers and documents.
- Slack/ WeChat / WhatsApp: Team communication and project management.

4. Reference Materials

- Research Papers and Documentation, used to guide model improvement and experimental design.
- Machine Learning and Deep Learning Books.
- Online Courses & Tutorials: Online learning resources such as Coursera, Udacity, YouTube, etc., for learning 3D object detection and point cloud processing technology.

5.3 Roles & Responsibilities

Roles and responsibilities of each member for the entire semester have been listed in the table below.

Name	Roles	Responsibilities
Lewis Liu	1. GitHub code management 2. YOLO v8, v11 model 3. Report Writing.	1. Manage the GitHub code repository for the project and update it with every change. 2. Implement and train YOLOv8 and YOLOv11 models, attempt to customize YOLOv11, and identify optimal parameters for our dataset. 3. Conduct experiments and performance evaluations of YOLO models, including hyperparameter tuning (e.g., k-fold cross-validation) to identify the best hyperparameters. 4. Write reports, including explanations of model principles and analysis of results. 5. Assist team members in integrating code and resolving technical issues related to model development.
Kaiyuan Xu		
Zechen Zhu		
Jing Luo		
Jiayuan Ma		
Taimur Enam	1. Data Sourcing 2. Data Analysis 3. Model Training 4. Report Writing	1. Was responsible for data collection and sourced publicly available 2D maritime radar datasets including MOANA, SAR-Ship and Dual Polarimetric SAR. 2. Wrote code in Python to conduct data analysis on the collected datasets to get insights and produced comparative analysis between three datasets. 3. Trained and evaluated the CA-CFAR model.

		4. Contributed in report writing including the Group Final Report, Group Proposal Report, Group Progress Report and all the Project Status Checking reports.

6. MILESTONES / SCHEDULE

Milestone	Tasks	Reporting	Date
Week-1	Project allocation and kick-off	Unit Coordinator	26-02-2025
Week-2	Understanding and discuss the project requirements and required data	Weekly tutorial	07-03-2025
Week-3	1. Collect publicly available radar datasets 2. Brief project requirements discussion with the client	Weekly tutorial presentation & Client emailing to review the work plan	13-03-2025
Week-4	1. Perform data analysis on WaterScene datasets 2. Explore different deep learning models and literature review 3. Write and submit Project Status Checking form	Client meeting to review the work plan & Weekly tutorial presentation	23-03-2025
Week-5	1. Write and submit Proposal Report 2. Training different deep learning models using WaterScence datasets 3. Collect hardware resources from the client	Visit client's office & Weekly tutorial presentation	30-03-2025
Week-6	1. search for 2D radar datasets due to changes in project scope	Client meeting to review work progress & Weekly tutorial presentation	03-04-2025

Week-7	1. Trained models using MOANA dataset	Client meeting to review models & Weekly tutorial presentation	10-04-2025
Week-8	1. Train models using MOANA dataset	Client meeting to review models & Weekly tutorial presentation	17-04-2025
Week-9	1. Train models using MOANA dataset 2. Fine tune and optimize training processes for better performing models 2. Write and submit Progress Report	Client meeting to review models performance & Weekly tutorial presentation	04-05-2025
Week-10	1. Train models using MOANA dataset 2. Fine tune and optimize training processes for better performing models 3. Prepare data analysis report	Weekly tutorial presentation	08-05-2025
Week-11	1. Evaluate the selected models performance 2. Fine tune and optimize training processes for better performing models 3. Prepare data analysis report	Client meeting to review models performance & Weekly tutorial presentation	15-05-2025
Week-12	1. Prepare presentation slides 2.Start writing the Final Report 3. Fine tune and optimize training processes for better performing models	Presentation & Weekly tutorial presentation	18-05-2025 (Updated due date for presentation submission: 06-06-2025)

Week-13	1. Continue working on the final report 2. Handover the code repository for models and datasets	Client meeting to showcase the project & Weekly tutorial presentation	25-05-2025 (Updated due date for final report submission: 13-06-2025)

7. RESULTS

7.1 Performance

Firstly, we listed the effects of all different models under different learning rates, Weight Decay and optimizer. We compared the mAP50 and mAP50-90 on the verification set, and the results as shown in the following table:

Model	Optimizer	Learning Rate	Epoch	Weight Decay	mAP50	mAP50-90
YOLOv11n	Adam	0.003	30	0.001	0.870	0.752
YOLOv11s	Adam	0.001	30	0.001	0.880	0.770
YOLOv8n	SGD	0.003	30	0.0001	0.880	0.749
YOLOv11n	SGD	0.005	30	0.001	0.883	0.750
FasterRCNN	SGD	0.005	8	0.0005	0.837	0.653
EfficientDet	Adam	0.001	56	0.0005	0.380	0.296
CA-CFAR	NA	NA	NA	NA	0.003	0.002

Table model result summarises the experimental results of various object detection models in this project, including YOLOv11n, YOLOv11s, YOLOv8n, FasterRCNN, EfficientDet, and CA-CFAR.

YOLOv11n(Adam optimiser): Achieved the best performance when using the Adam optimiser with a learning rate of 0.003, with mAP50 reaching 0.870 and mAP50-90 reaching 0.752. The model demonstrated relatively stable detection capabilities across different target sizes. Since n is the smallest YOLO framework, but the dataset is 2048x2048, the framework is too small for this dataset as well.

YOLOv11s(Adam optimiser): model with the best overall performance in this experiment. When using the Adam optimiser with a learning rate of 0.001, mAP50 reaches 0.880 and mAP50-90 reaches 0.770. As a medium-sized model in the YOLOv11 series, it achieves a good balance between detection accuracy and speed. It

performs optimally on the current dataset and is suitable for the majority of object detection scenarios.

YOLOv8n (SGD optimiser): When using the SGD optimiser with a learning rate of 0.003, YOLOv8n achieves an mAP50 of 0.880 and an mAP50-90 of 0.749, performing on par with the latest YOLOv11 series models and demonstrating high detection accuracy.

YOLOv11n (SGD optimiser):When using the SGD optimiser with a learning rate of 0.005, YOLOv11n achieves an mAP50 of 0.883 and an mAP50-90 of 0.750, showing a slight improvement over the Adam optimiser and demonstrating superior detection performance.

FasterRCNN (SGD optimiser):When using the SGD optimiser with a learning rate of 0.005, FasterRCNN achieves an mAP50 of 0.837 and an mAP50-90 of 0.653. Although its overall performance is slightly lower than that of the YOLO series models, as a classic baseline method in the field of object detection, it still performs stably.

EfficientDet (Adam optimiser):Using the Adam optimiser with a learning rate of 0.001 and trained for 96 epochs, achieves an mAP50 of only 0.380 and an mAP50-90 of 0.296, indicating that the model has poor adaptability and performs poorly on the current dataset.

CA-CFAR: As a model trained with a traditional algorithm, CA-CFAR (Cell-Averaging Constant False Alarm Rate) does not include learning-based optimization and is not adaptable to complex background clutter and object variability. As we see from the results, it performs very poorly with mAP50 being as low as 0.003 and mAP50-90 being as low as 0.002.

7.2 Comparison Methods

As can be seen from the table, both YOLOv11n and YOLOv8n demonstrate excellent accuracy. Considering model size and inference speed, YOLOv11n is more suitable for practical deployment scenarios. However, FasterRCNN and EfficientDet lag behind the YOLO series in terms of parameter count and inference speed, and struggle to fully leverage their advantages at high resolutions. Additionally, CA-CFAR performs extremely poorly compared to other models.

While YOLOv11n and YOLOv8n have similar detection accuracy, YOLOv11s achieves the best overall performance on the current dataset. Although YOLOv11s has slightly more parameters, it demonstrates a more significant improvement in accuracy, making it the selected detection model for this project.

8. DISCUSSION

In this section, discuss the results, implications and significance of your project contributions. The implication should be explained in more detail in the final report than your initial proposal. This section is also where you state how your findings contribute to existing gaps in the field or recommendations – practical suggestions to implementation of findings/outcomes.

9. LIMITATIONS AND FUTURE WORKS

Although a number of object detection models were successfully implemented and evaluated, the project ran into a number of obstacles and restrictions that affected the models' overall performance as well as the development process. The high computational cost of deep learning model training was one of the main challenges. Significant time and processing power were needed for training, particularly when working with high-resolution image data. In reality, using such images often resulted in system slowdowns or memory overflow errors, which interrupted or prolonged the training process. The range of experiments and the complexity of models that might be investigated were both constrained by these hardware limitations.

Apart from computational constraints, the training and evaluation dataset posed a significant obstacle. In terms of item classes, backgrounds, and environmental conditions, it was somewhat small and lacked diversity. As a result, the models have trouble successfully generalizing to novel or intricate situations. Therefore, even while certain models did well on the test data, their performance could not transfer well to real-world applications where variability is far higher.

Another problem was that some models had a propensity to plateau in performance somewhat early in the training process. Faster R-CNN, for example,

demonstrated very modest accuracy improvements even after several iterations of hyperparameter adjustment. This shows the model's potential for improvement was limited, maybe as a result of both computing bottlenecks and dataset constraints. In addition, a significant amount of the project schedule was devoted to data preprocessing. It took a lot of time to convert annotation files from XML format into the format needed by training processes. Progress was further hampered by the fact that manual formatting and correction were frequently required.

Another difficulty that arose during the project was time limits. Exploring a larger variety of models and conducting comprehensive hyperparameter tuning for every model evaluated were not practicable due to the time constraints. Because of this, some models might not have realized their full potential because of limited tuning and experimentation, even though the research offered helpful comparative insights.

Looking ahead, there are a number of encouraging avenues to enhance object detection algorithms' resilience and efficacy. Increasing the size and variety of the training dataset is an important step. Increasing the quantity of annotated photos with different backgrounds, occlusions, lighting conditions, and item classifications might aid models in learning more generalizable characteristics and lessen overfitting. More sophisticated models might be trained in a reasonable amount of time and memory by utilizing cloud-based GPUs or model compression techniques, which could assist get around hardware limits.

Moreover, investigating more recent model architectures like ensemble methods and transformer-based detectors may result in even greater accuracy and versatility. These cutting-edge models have performed well in recent experiments and may be useful additions to those conducted in the future. Finally, models should be tested in realistic settings with different lighting, motion blur, and occlusion. A deeper comprehension of model reliability in real-world applications would be provided by testing in such circumstances. Combining detection with tracking may also be advantageous, particularly for applications involving video where it's crucial to preserve item identities across time.

REFERENCES