Ship Object Detection: Final Report

Kerem Adalı

June 9, 2025

1 Introduction

This Final report outlines the approach for the Ship Object Detection project as part of the Introduction to Artificial Intelligence course (Summer 2025). The objective is to develop a solution for detecting ships and boats from aerial images, which falls under the category of object detection tasks in computer vision.

2 Dataset Description

2.1 Overview

The dataset[1] consists of aerial images containing ships and boats of various sizes, types, and orientations. Based on initial exploration, the following observations can be made:

2.2 Descriptive Statistics

- **Image count:** 661 images in the dataset
- **Image resolution:** Varies, but typically around 500x500 pixels
- **Color format:** RGB (3 channels)
- **Annotation format:** PASCAL VOC format
- **Object categories:** Single class ("boat")
- **Objects per image:** Ranges from 1 to 15+ ships per image

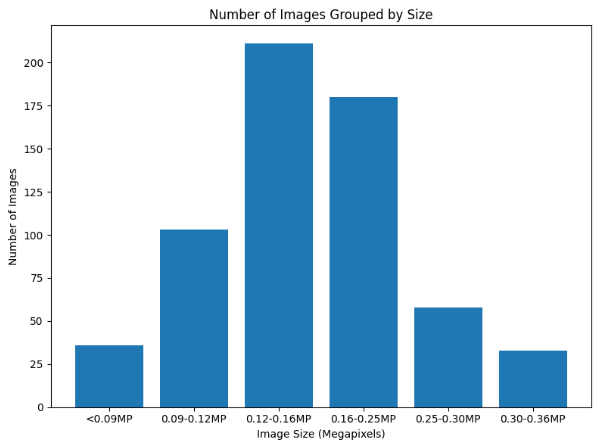
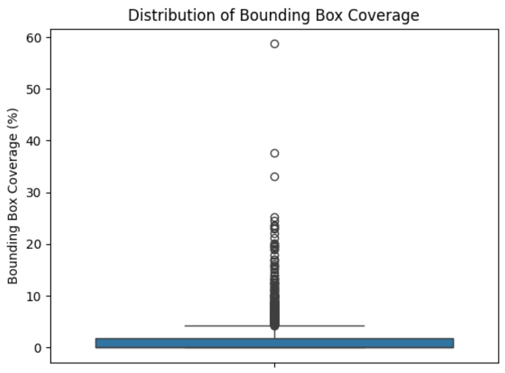


Figure 1: Number of Images grouped by size

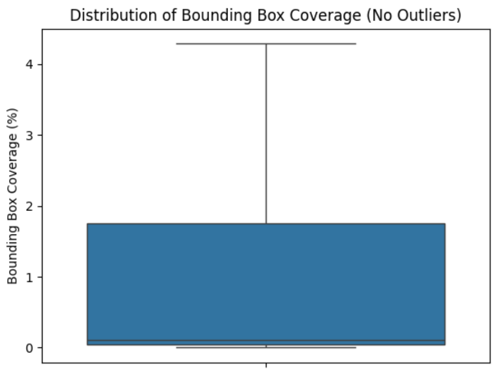
2.3 Sample Visualizations

Initial visual analysis reveals several interesting patterns:

- Ships vary in size, from small boats to large vessels
- Images contain diverse backgrounds including open water, harbors, and coastal areas
- Some images contain tightly clustered ships (port scenarios)
- Varying angles of capture (directly overhead vs. angled aerial views)
- Each ship bounding box covers approximately 2-4% of the image for the majority of the time, although having outliers



((a)) Bounding box coverages, all ships



((b)) Bounding box coverages, without outliers

Figure 2: Bounding box coverages of ships on the image

2.4 Challenges Identified

- Small objects: Some ships appear very small in images taken from high altitude
- Occlusion: Ships in harbors may partially occlude each other
- Background confusion: Some coastal structures or waves may resemble ships
- Varying lighting conditions: Reflections on water surfaces
- Different orientations: Ships appear at various angles

3 Proposed Approach

3.1 Dataset Splitting

The dataset will be split as follows:

- Training set: 70% of the data (approximately 435 images)
- Validation set: 15% of the data (approximately 93 images)
- Test set: 15% of the data (approximately 93 images)

The splitting will be performed using random sampling to ensure a balanced distribution of scenarios (open water, harbor, etc.) and ship densities across all sets.

3.2 Selected Algorithms and Models

For this project, I plan to implement and compare multiple object detection algorithms:

3.2.1 YOLOv8

YOLOv8[4] is the probably the most famous in the YOLO (You Only Look Once) family, known for its excellent balance between speed and accuracy. Usually used for real-time solutions.

3.2.2 RetinaNet with ResNet50 backbone

RetinaNet model with ResNet50 backbone fine-tuned on COCO in 800x800 resolution with FPN features created from P5 level.

3.2.3 DETR model with ResNet-50 backbone

DETR (DEtection TRansformer)[2] is an end-to-end object detection architecture that uses a transformer encoder-decoder with a ResNet-50 backbone.

3.3 Implementation Tools and Framework

- **Primary framework:** PyTorch with Torchvision
- **Object detection libraries:**
  - Ultralytics for YOLOv8 or Keras with YOLOv8 backbone
  - Pytorch implementation from yhenon's[3] github repo RetinaNet with ResNet50 bacbkbone
  - Pytorch for DETR
- **Data preprocessing:** OpenCV, pytorch
- **Evaluation metrics:** pycocotools
- **Visualization:** OpenCV, Matplotlib, seaborn
- **Training environment:** My computer with NVIDIA RTX 4050 Laptop GPU (45W), no notebooks
- **Version control:** GitHub

3.4 Data Preprocessing Pipeline

- **Image normalization:** Standard scaling (mean subtraction and division by standard deviation)
- **Data augmentation:**
  - Random horizontal and vertical flips
  - Random rotation ($\pm 15°$)
  - Random resizing (inside predefined borders with min and max values)
  - Random cropping (maintaining all objects)
- **Image resizing:** Resize to 640×640 for YOLOv8, 800×800 for other models

4 Evaluation Methods

4.1 Performance Metrics

- **Primary metrics:**
  - Mean Average Precision (mAP) at IoU thresholds of 0.5 (mAP@0.5)
  - Mean Average Precision (mAP) at IoU thresholds from 0.5 to 0.95 with a step of 0.05 (mAP@0.5:0.95)
- **Secondary metrics:**
  - Inference speed (ms or FPS)

4.2 Model Comparison

The models will be compared across the following dimensions:

- Detection accuracy (using the metrics defined above)
- Performance across different ship sizes (small, medium, large)
- Computational efficiency (inference time etc.)

5 Experimental Setup

5.1 Hyperparameter Tuning

For each model, the following hyperparameters will be explored:

- Learning rate (initial and scheduling)
- Batch size
- Number of training epochs

5.2 Training Strategy

- Initial training with default parameters
- Performance analysis
- Iterative hyperparameter tuning
- Final model training with optimal parameters

6 DETR with ResNet-50 Backbone

6.1 Data Preparation

- **VOC→COCO conversion**: Forked and modified the `voc2coco` repository to convert PASCAL VOC annotations into COCO format. All "boat" annotations were preserved and remapped to a single class ID (0) for direct compatibility with DETR's data loader.

6.2 Model Modifications

- **Custom single-class DETR**: Forked the official DETR repository and modify it to make it suitable for custom dataset training, removed the default classification head's weight and bias tensors (originally sized for 91 COCO classes), and re-initialized it to output a single "boat" class plus the "no-object" token. All other pretrained weights (backbone, transformer encoder/decoder) were left intact.

6.3 Training Setup

Environment

- **Hardware**: NVIDIA RTX 4050 Laptop GPU, 6 GB VRAM
- **Framework**: PyTorch 2.3.1+cu121,

Parameter	Value
Batch size	2
Optimizer	AdamW (lr $= 1 \times 10^{-4}$)
LR scheduler (StepLR)	step_size $= 50$, $\gamma = 0.59$
Total epochs	300
Weight decay	1×10^{-4}
Classification Loss function	Cross Entropy
Bounding box Loss function	L1 (Mean Absolute Error)

Table 1: Modified training hyperparameters for DETR with ResNet-50

Hyperparameters

Data Augmentation

- **Normalization**: Normalize the RGB channels with ImageNet means $[0.485, 0.456, 0.406]$, stds $[0.229, 0.224, 0.225]$.
- **Spatial transforms**: random horizontal flip ($p = 0.5$), random resize the smaller side to 480–800 px (bigger side max 1333px), random crop.

6.4 Evaluation Metrics

- **Primary**: mAP@0.5, mAP@[0.5:0.95]
- **Secondary**: Inference time

6.5 Evaluation Results on the test set

Metric	area	Result
mAP@0.5	all	**83.3%**
mAP@[0.5:0.95]	all	**55.0%**
mAP@[0.5:0.95]	small	**19.2%**
mAP@[0.5:0.95]	medium	**77.5%**
mAP@[0.5:0.95]	large	**90.1%**
Inference speed	**30-45 ms/image**	

Table 2: Evaluation results for DETR (ResNet-50) with modified training hyperparameters (my fork)

Metric	area	Result
mAP@0.5	all	**81.15%**
mAP@[0.5:0.95]	all	**53.7%**
mAP@[0.5:0.95]	small	**18.8%**
mAP@[0.5:0.95]	medium	**76.6%**
mAP@[0.5:0.95]	large	**85.8%**
Inference speed	**30-45 ms/image**	

Table 3: Evaluation results for DETR (ResNet-50) (official repo)

6.6 Example detections on the test set

Below is some examples of DETR predictions of specific group of images.

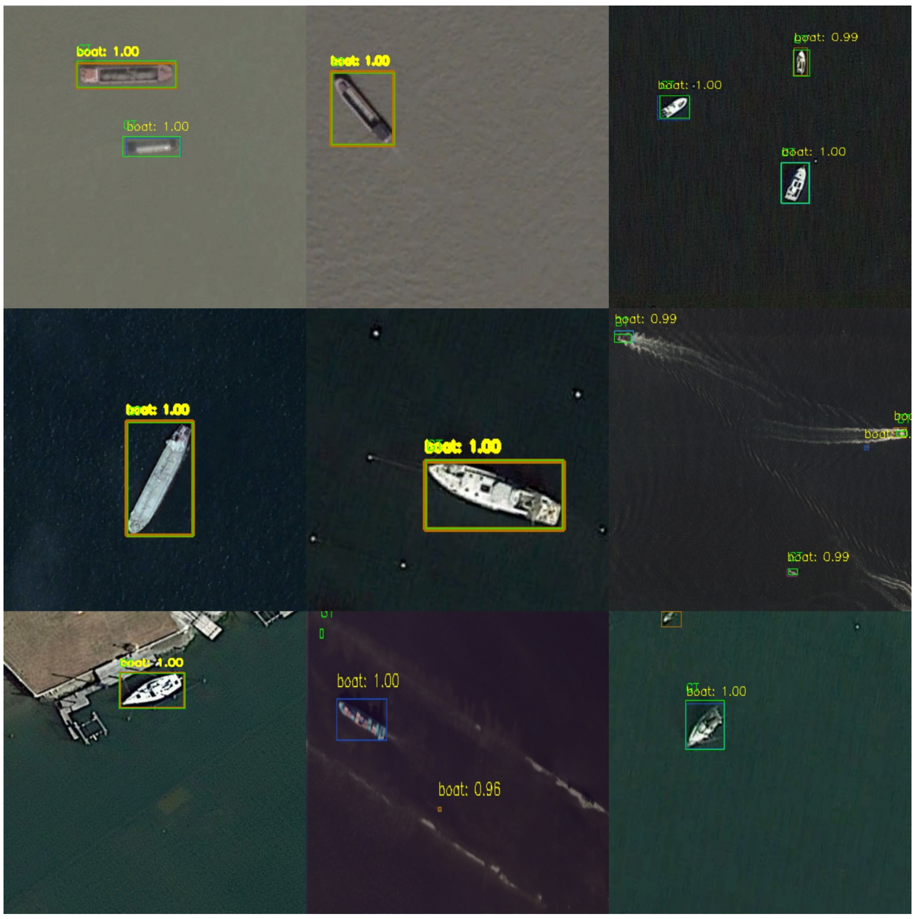
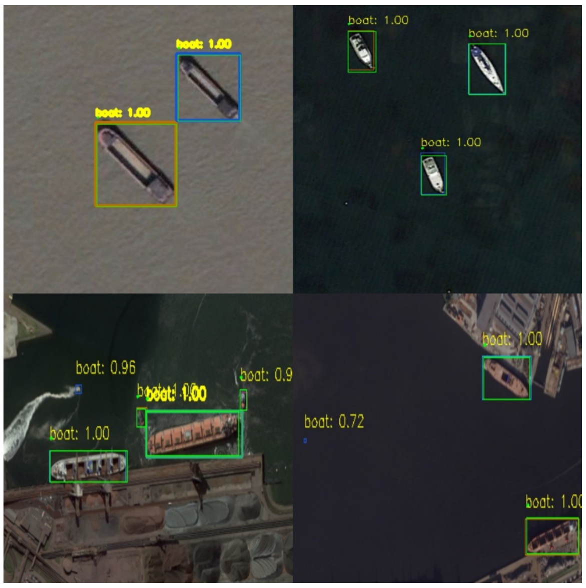
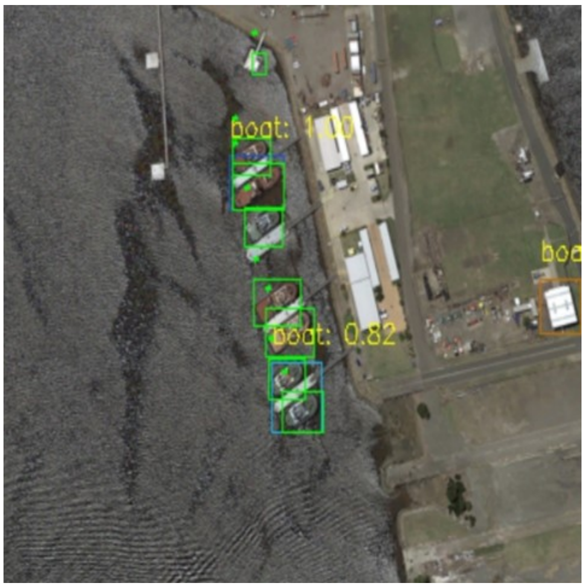
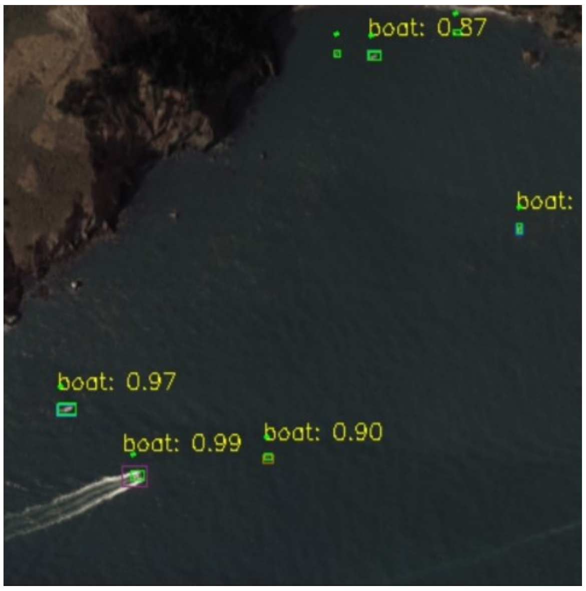
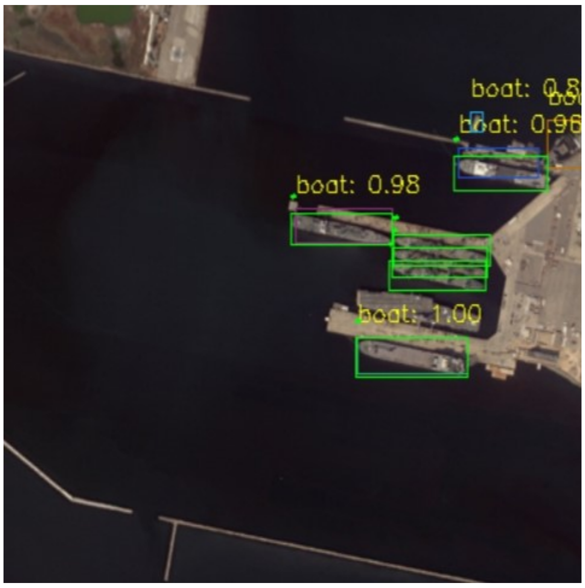


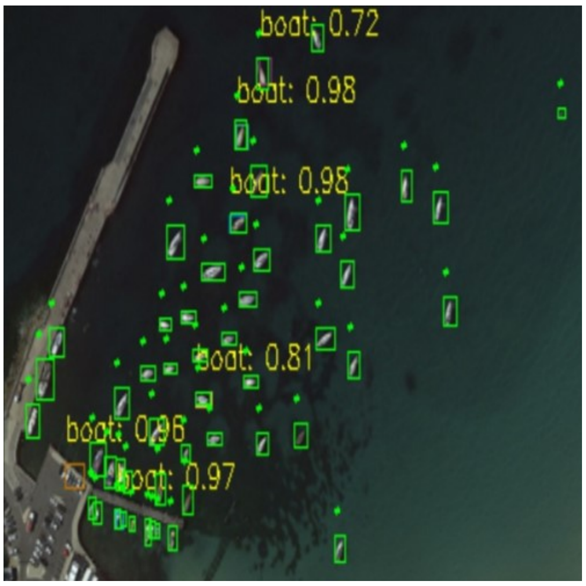
Figure 3: 9 Random images predicted by DETR


((a)) Multiple boats


((b)) Multiple small boats clustered docked in port


((c)) Small, distant scattered boat detection


((d)) Multiple medium sized boats clustered docked in port


((e)) Many extremely small boats

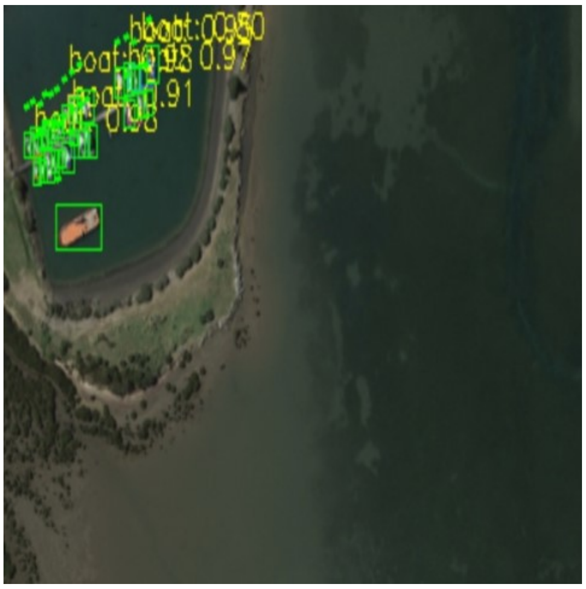

((f)) Many extremely small boats

Figure 4: Sample DETR detections on the test set. Yellow and blue bounding boxes indicate the predicted "boat" class along with confidence scores. Green bounding boxes indicate the ground truth.

6.6.1 What we can deduce from these predictions

- The model performs very well with a clear sight of ships in the image.
- The model struggles to predict all of the ground truth correctly when the ground truth amount for a single image is too many for it to handle.
  How can we explain it?: The DETR model runs 100 queries for a single detection, only the ones with a confidence over the threshold is drawn, these 100 queries generally can not detect more than a certain amount of ground truth.
- The model doesn't struggle predicting small ground truth when they are scattered and scarce around the image, proving the explanation for the latter item.
- The model struggles with detecting the ground truth docked near the harbor/marina, especially when the image of a marina resembles an image of a ship. And might predict as a false positive.
- The model might predict ground (non water) targets as ships, thus a false positive.

6.7 Experiment Analysis

- **Scheduler tweak**: Reducing `step_size` from 200 to 50 and increasing γ from 0.1 to 0.59 yielded a final +2.7% gain in mAP@0.5 compared to the default scheduler.
- **Single-class head**: Re-initializing the classification head enabled rapid adaptation to the "boat" class, reducing confusion with background tokens.
- **What didn't work?**: Although increasing the mAP@0.5:0.95 on small sized bounding boxes by +2% having only mAP@0.5:0.95 18.8% suggests a different approach should be taken towards the detection of small boats or faraway shots.

6.8 Where the model struggled

- **Small sized boats**: The model struggled with detecting small or boats from very far away, having only mAP@0.5:0.95 on small area bounding boxes 18.6%. Suggesting a different approach should be taken towards the detection of small boats or faraway shots.
  What can be done?: a sliding window technique which we can describe as, dividing each photo into 4-8 sections (or more) and running inference through all of them to have the small sized targets appear bigger. Cons of this approach is it requires more computing power and requires faster processing unit, whether it be GPU, CPU or TPU etc.
- **Boats parked near coast or very close to eachother:** The model struggled distinguishing marina spaces, docks from boats and sometimes detected parts of land as boats. This is called a false positive which suggests the possibility of overfitting.
  What can be done?: More data with mentioned circumstances should be gathered for the training.

7 Retinanet with ResNet-50 Backbone

7.1 Data Preparation

- **VOC→COCO conversion**: Forked and modified the `voc2coco` repository to convert PASCAL VOC annotations into COCO format.

7.2 Training Setup

Environment

- **Hardware**: NVIDIA RTX 4050 Laptop GPU, 6 GB VRAM
- **Framework**: PyTorch 2.3.1+cu121

Parameter	Value
Batch size	2
Optimizer	Adam (lr = 1×10^{-5})
LR scheduler (ReduceLROnPlateau)	patience = 3, $\gamma = 0.1$
Total epochs	106
Weight decay	0
Loss function	Focal Loss (a modification of Cross Entropy)

Table 4: Training hyperparameters for RetinaNet with ResNet-50

Hyperparameters

Data Augmentation

- **Normalization**: Normalize the RGB channels with ImageNet means $[0.485, 0.456, 0.406]$, stds $[0.229, 0.224, 0.225]$.
- **Spatial transforms**: random horizontal flip ($p = 0.5$), random resize (bigger side max 1024px and minimum 608px), random crop.

7.3 Evaluation Metrics

- **Primary**: mAP@0.5, mAP@[0.5:0.95]
- **Secondary**: Inference time

7.4 Evaluation Results on the test set

Metric	area	Result
mAP@0.5	all	**77.7%**
mAP@[0.5:0.95]	all	**51.5%**
mAP@[0.5:0.95]	small	**19.1%**
mAP@[0.5:0.95]	medium	**73.1%**
mAP@[0.5:0.95]	large	**83.6%**
Inference speed	**45-60 ms/image**	

Table 5: Evaluation results for RetinaNet (ResNet-50))

7.5 Example detections on the test set

Below is some examples of RetinaNet predictions of specific group of images.

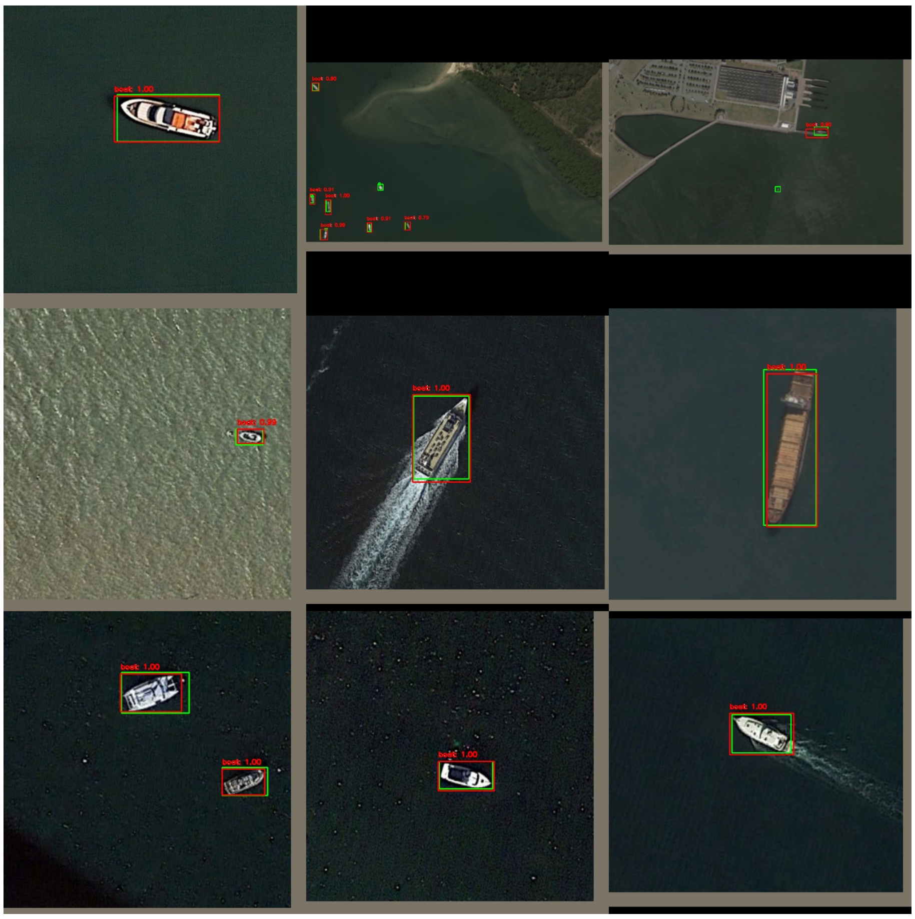
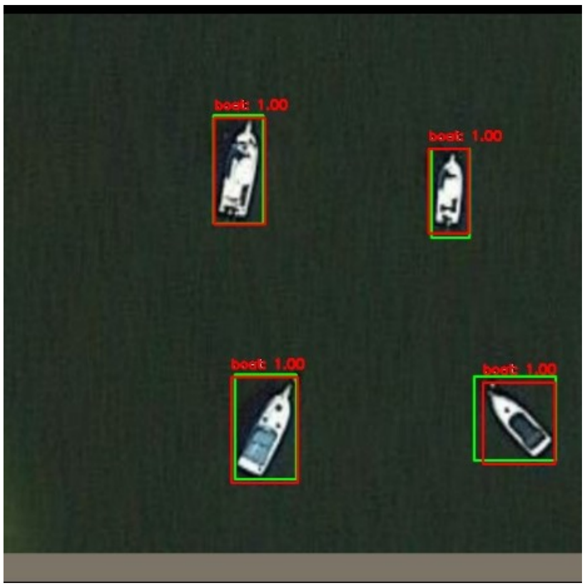
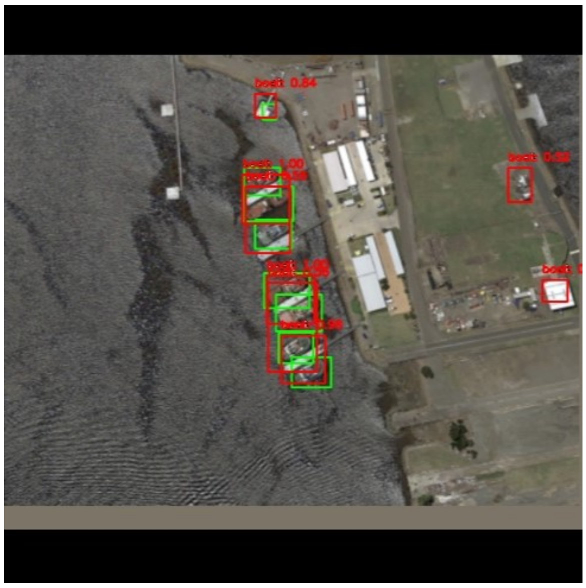
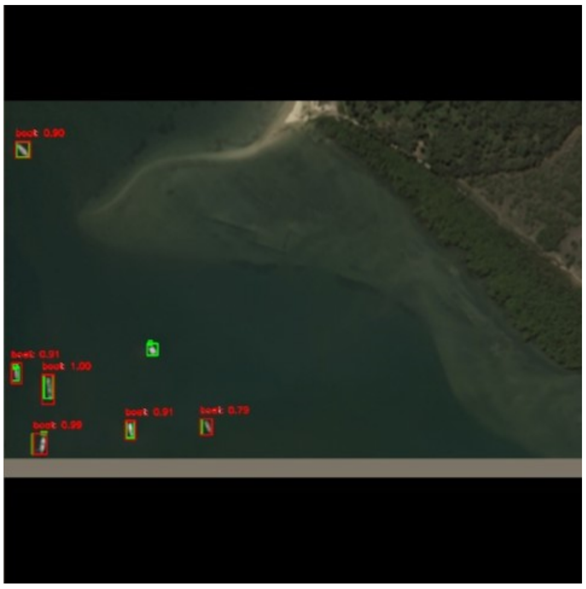


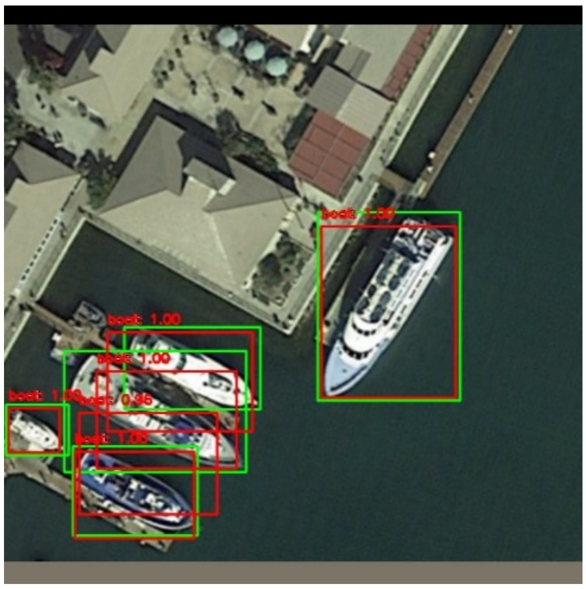
Figure 5: 9 Random images predicted by RetinaNet

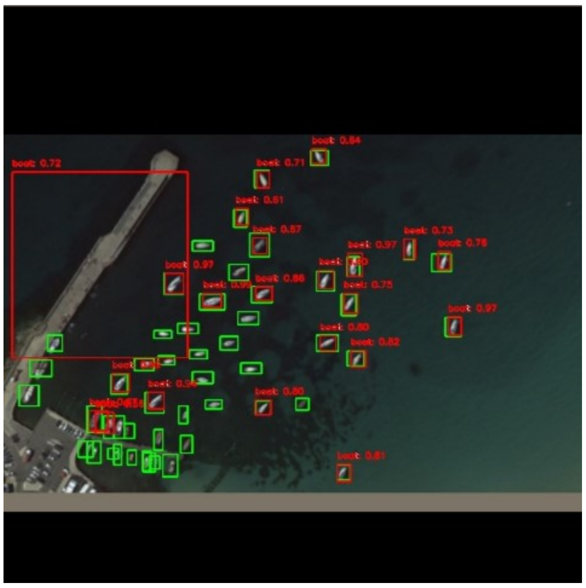


((a)) Multiple boats


((b)) Multiple small boats clustered docked in port


((c)) Small, distant scattered boat detection


((d)) Multiple medium sized boats clustered docked in marina


((e)) Many extremely small boats

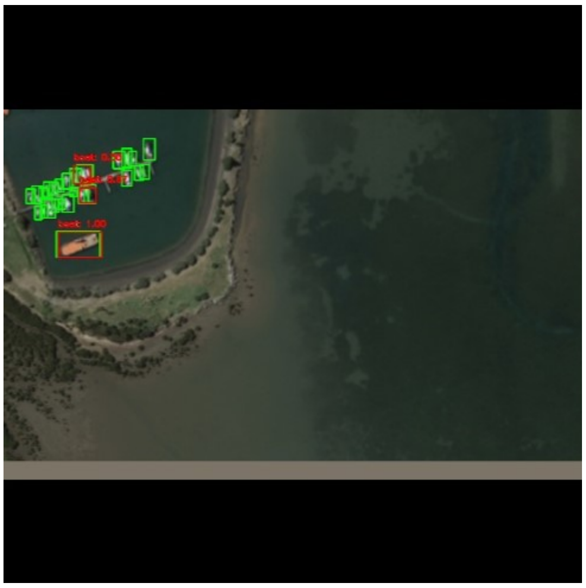

((f)) Many extremely and dense small boats

Figure 6: Sample RetinaNet detections on the test set. Red bounding boxes indicate the predicted “boat” class along with confidence scores. Green bounding boxes indicate the ground truth.

7.5.1 What We Can Deduce from These Predictions

- RetinaNet achieves very high performance on medium and large boats, with mAP@[0.5:0.95] for medium at 73.1% and for large at 83.6%.
- The model detects well when boats are isolated and clearly visible—e.g., single vessels on open water are consistently localized.
- RetinaNet struggles significantly with small boats: mAP@[0.5:0.95] on small area bounding boxes is only 19.1%.
- When multiple boats cluster tightly (e.g., in a marina or dock), the model sometimes merges adjacent boats or confuses dock structures with vessels, leading to missed detections or false positives.
- False positives occasionally occur on non-water regions that resemble boat shapes (e.g., docks, buoys), indicating that background texture can still mislead the single-scale anchor priors.

7.6 Experiment Analysis

- **Batch size and learning rate:** Using batch size = 2 and Adam at 1×10^{-5} yielded stable convergence but limited gradient diversity. A larger batch (if VRAM allowed) might further improve mAP, especially on small objects.
- **LR scheduler (ReduceLROnPlateau):** With patience = 3 and $\gamma = 0.1$, the scheduler reduced the learning rate at plateau and helped avoid over-fitting in later epochs. Performance on medium and large boats benefited most from this schedule.
- **Data augmentation:** Random horizontal flips, multi-scale resizing (max side = 1024 px, min side = 608 px), and random crops improved generalization to varied boat orientations and scales. However, these augmentations had only a modest impact on small-boat detection.
- **What didn't work:** Despite data augmentation, small-boat mAP@[0.5:0.95] remained under 20%. This suggests that a single-pass, anchor-based head is insufficient for very small or distant targets without further architectural changes (e.g., feature pyramid or cascade).

7.7 Where the Model Struggled

- **Small-sized boats:** With only 19.1% mAP@[0.5:0.95] on small areas, RetinaNet fails to reliably detect boats occupying few pixels. Improving this would require higher-resolution feature maps or specialized small-object sub-heads.
- **Tightly clustered boats in marinas:** Boats docked close together or near piers often get merged into a single box or confused with dock infrastructure, producing both false negatives and false positives. This indicates the need for better anchor stratification or post-processing (e.g., stricter NMS thresholds) for dense scenes.
- **False positives on docked/coastal structures:** Certain harbor backgrounds (ramps, buoys, breakwaters) can resemble boat outlines. The model occasionally misclassifies these as "boat," suggesting overfitting to texture patterns rather than shape. Additional negative samples (e.g., docks without boats) could help.

8 YOLOv8 (Medium model)

8.1 Data Preparation

- **COCO→YOLO conversion**: Wrote a simple script that creates YOLO type labels from the COCO JSON files.

8.2 Training Setup

Environment

- **Hardware**: NVIDIA RTX 4050 Laptop GPU, 6 GB VRAM
- **Framework**: ultralytics, PyTorch 2.3.1+cu121

Parameter	Value
Batch size	Auto (to take 60 percent of the VRAM)
Optimizer	AdamW (lr = 0.002)
LR scheduler (LambdaLR)	Linear
Total epochs	150
Weight decay	0.005
Loss functions	Box Loss, Class Loss, Distributed Focal Loss

Table 6: Training hyperparameters for YOLOv8

Hyperparameters

Data Augmentation

- **Mosaic:** Combine four random images into one, with probability $p =$ hyp.mosaic, to enrich context and object scale variation.
- **Random Perspective:** Apply random rotation ($\pm$hyp.degrees°), translation ($\pm$hyp.translate $\times$ image size), scaling (factor hyp.scale), shearing ($\pm$hyp.shear°), and perspective warp (hyp.perspective), with optional LetterBox resizing if `stretch=False`.
- **Copy–Paste:** With probability $p =$ hyp.copy_paste, copy objects from one image and paste into another, using either a flipped or mosaic–affine pre-transform depending on `hyp.copy_paste_mode`.
- **MixUp:** Blend two images and their labels, with probability $p =$ hyp.mixup, to improve robustness to occlusion and background variation.
- **CutMix:** Cut a patch from one image and paste into another, with probability $p =$ hyp.cutmix, promoting learning of partial object appearances.
- **Albumentations:** Apply a suite of photometric and geometric augmentations (brightness, contrast, blur, etc.) with probability 1.0 for diverse visual perturbations.
- **Random HSV:** Perturb hue ($h_{\text{gain}} =$ hyp.hsv_h), saturation ($s_{\text{gain}} =$ hyp.hsv_s), and value ($v_{\text{gain}} =$ hyp.hsv_v) channels for color jitter.
- **Random Flip (vertical):** Flip images vertically with probability $p =$ hyp.flipud.
- **Random Flip (horizontal):** Flip images horizontally with probability $p =$ hyp.fliplr, using `flip_idx` for keypoint consistency if available.

8.3 Evaluation Metrics

- **Primary**: mAP@0.5, mAP@[0.5:0.95]
- **Secondary**: Inference time

8.4 Evaluation Results on the test set

Metric	area	Result
mAP@0.5	all	**90.6%**
mAP@[0.5:0.95]	all	**61.0%**
mAP@[0.5:0.95]	small	**31.5%**
mAP@[0.5:0.95]	medium	**80.0%**
mAP@[0.5:0.95]	large	**87.5%**
Inference speed		**36 ms/image**

Table 7: Evaluation results for YOLOv8s

8.5 Example detections on the test set

Below is some examples of YOLOv8 predictions of specific group of images.

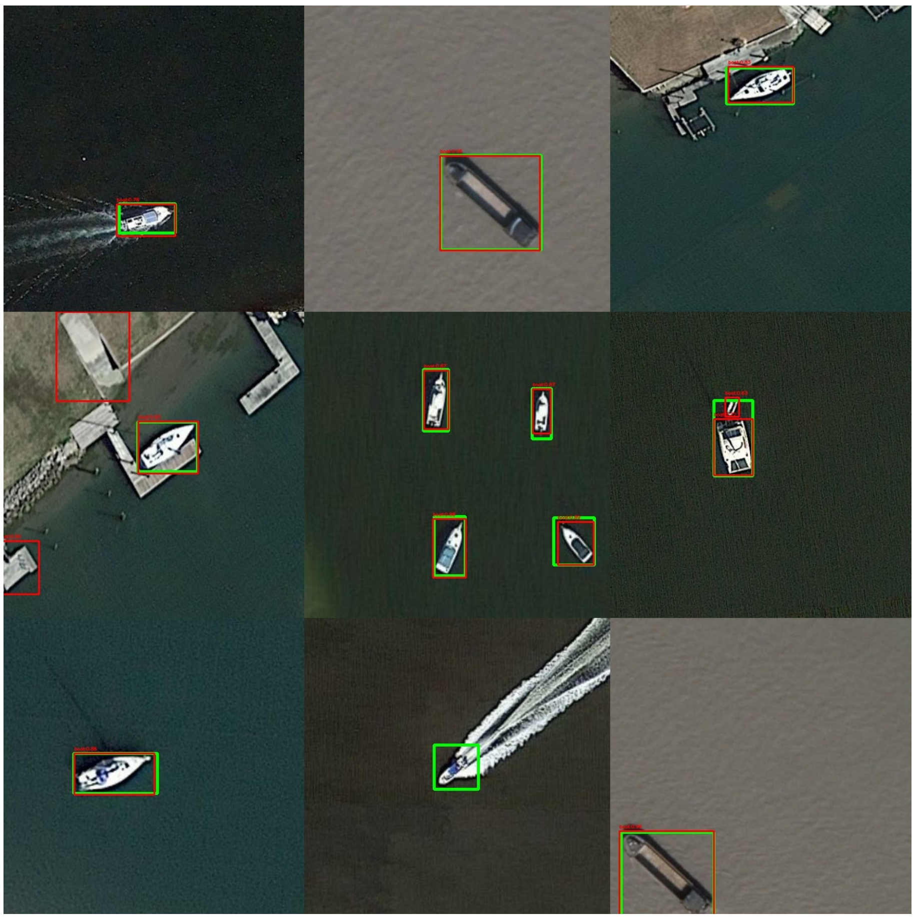
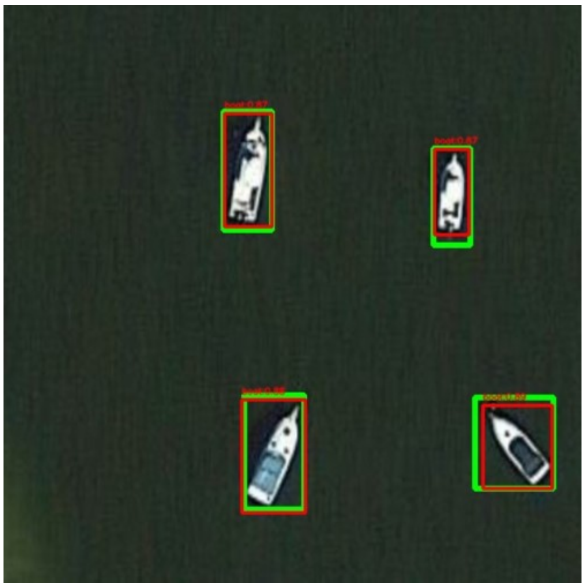
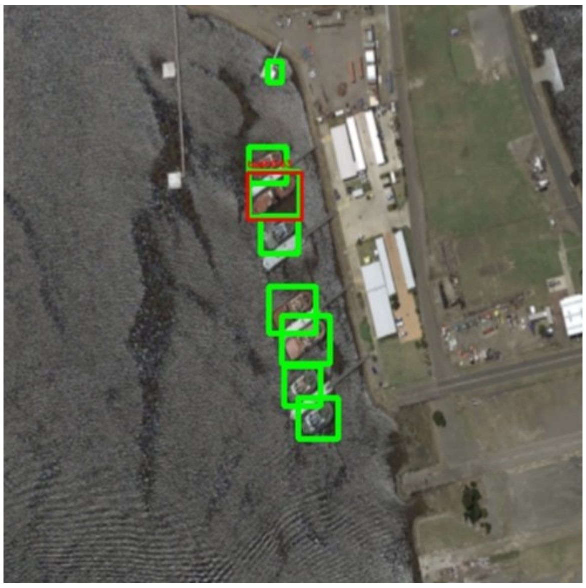


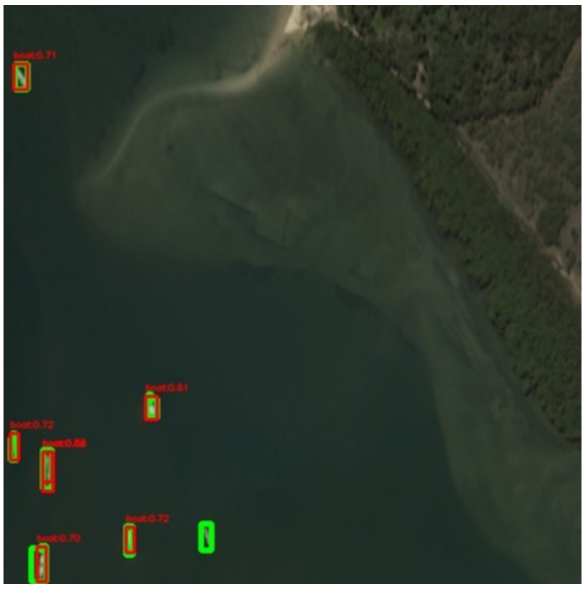
Figure 7: 9 Random images predicted by YOLOV8



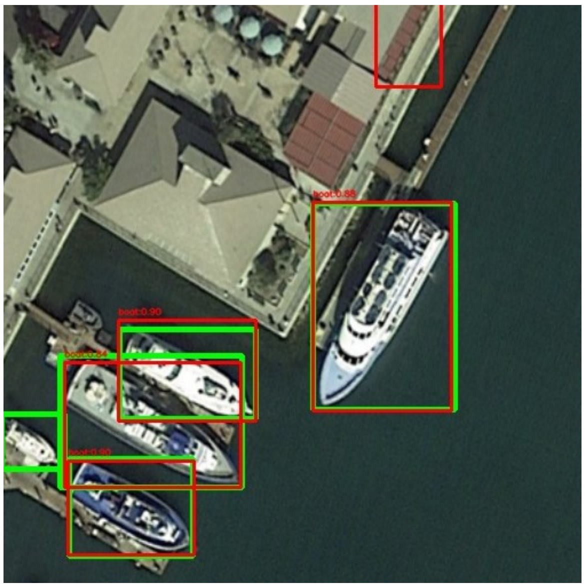
((a)) Multiple boats



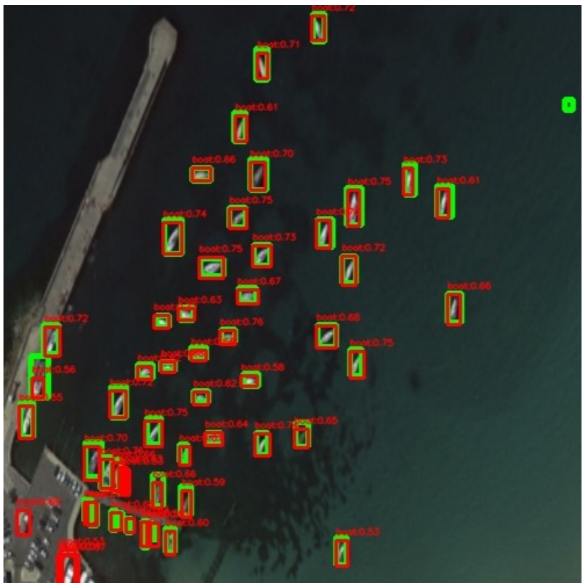
((b)) Multiple small boats clustered docked in port



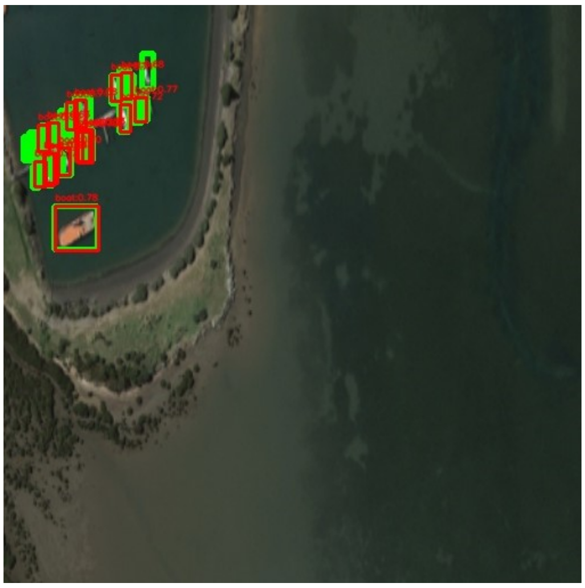
((c)) Small, distant scattered boat detection



((d)) Multiple medium sized boats clustered docked in marina



((e)) Many extremely small boats



((f)) Many extremely and dense small boats

Figure 8: Sample YOLOV8 detections on the test set. Red bounding boxes indicate the predicted "boat" class along with confidence scores. Green bounding boxes indicate the ground truth.

8.5.1 What We Can Deduce from These Predictions

- **Small-object gains:** The YOLOv8 Medium model markedly improves detection of extremely small boats (mAP@[0.5:0.95] small = 31.5%), outperforming both RetinaNet and DETR on tiny targets.
- **Strong overall performance:** High precision on medium (80.0%) and large (87.5%) boats, with mAP@0.5 reaching 90.6%.
- **Background confusion persists:** Some false positives on ground-like structures (docks, buoys) remain.
- **Shoreline challenges:** Boats near the shore are occasionally mislocalized or split between water and land.
- **Rare large-boat misses:** Very large targets are almost always detected, but on rare occasions the model fails to register them.

8.6 Experiment Analysis

- **Model capacity:** Upgrading to the Medium backbone provided richer features, boosting small-object recall and overall mAP.
- **Optimizer & scheduler:** AdamW with a linear LR decay over 150 epochs yielded stable convergence and fine-tuned large-object boundaries.
- **Auto batch sizing:** Dynamically sizing the batch to 60% of VRAM improved gradient diversity without memory overflow.
- **Augmentation impact:** Mosaic, Copy–Paste, MixUp and CutMix had the greatest effect on tiny and occluded boats.

8.7 Where the Model Struggled

- **Ground-object false positives:** Docks, buoys and rocky shorelines are sometimes misclassified as boats.
- **Shoreline proximity:** Boats hugging the shore can be split or only partially detected due to mixed background semantics.
- **Occluded clusters:** Tightly packed boats under piers still lead to merges or missed detections.
- **Rare large-boat misses:** On very large vessels spanning the frame, the model occasionally fails to propose any box.

9 Model Comparison

Table 8 summarizes all models on the same test split.

Model	mAP@0.5	mAP@0.5:0.95	Small	Medium	Large	Speed
RetinaNet	77.7%	51.5%	19.1%	73.1%	83.6%	45–60 ms/img
DETR	83.3%	55.0%	19.2%	77.5%	90.1%	30–45 ms/img
YOLOv8m	90.6%	61.0%	31.5%	80.0%	87.5%	36 ms/img

Table 8: Comparison of detection performance and speed

10 Conclusion

In this Final report, we compared three object detection models—RetinaNet, DETR, and YOLOv8—on aerial images of ships. Here are the main takeaways:

- YOLOv8 Medium achieves the highest overall accuracy (mAP@0.5:0.95 of 61.0%) and excels in detecting very small vessels, showing a 31.5% mAP on small objects, making it the strongest candidate for real-time, fine-grained ship detection.
- The modified DETR model offers a favorable trade-off between accuracy and inference speed (30–45 ms/image), outperforming RetinaNet on large objects (90.1% mAP@0.5:0.95) and maintaining robust performance across medium targets.
- RetinaNet remains competitive on medium and large ships (73.1% and 83.6% mAP@0.5:0.95, respectively) but is constrained by slower inference (45–60 ms/image) and limited small-object sensitivity.

Choosing the right model depends on the task: use YOLOv8 for quick, detailed detection; DETR for a balance of speed and accuracy; and RetinaNet when medium-to-large ships are the main focus.

For the next steps, to improve the usage of these models on real world applications these can be done:

- Using sliding-window inference to boost recall on tiny, distant boats.
- Gathering more images of crowded docks and shorelines to reduce false positives.

References

[1] Ships dataset.

[2] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In *European Conference on Computer Vision*, pages 213–229. Springer, 2020.

[3] Yann Henon. Pytorch implementation of retinanet object detection.

[4] Glenn Jocher, Ayush Chaurasia, and Jing Qiu. Ultralytics yolov8, 2023.