Report

Prepared by
Sujeet Kumar Singh
Assigned by Aethrone Aerospace Pvt. Limited

May 17, 2023

Multi-Class Object Detection on Aerial Maritime Datasets Using YoloV5 Model

1 Introduction

The purpose of this report is to present the findings and results of the aerial maritime object detection dataset, which was created using the YOLO V5 PyTorch framework. The dataset consists of 74 aerial photographs taken by a Mavic Air 2 drone, capturing various maritime scenes. The dataset includes 1,151 bounding boxes, encompassing different objects such as docks, boats, lifts, jetskis, and cars. The primary objective of this dataset is to perform aerial object detection tasks related to maritime environments.
The dataset comprises high-resolution aerial images taken by a quadcopter drone flying at an altitude of 400 ft. Each image has been manually labelled using the labelimg tool to create bounding boxes around the objects of interest. The labels have been categorized into multiple classes, including docks, boats, lifts, jetskis, and cars. This multi-class dataset provides a comprehensive set of objects commonly found in maritime settings.

1.1 Datasets Preparation

Once we have labeled our images using an annotation tool, we need to export the labels in YOLO format. Each image should have a corresponding *.txt file (unless there are no objects in the image). The format of the *.txt file is as follows:
Each object is represented by a separate row. The row contains information about the class, x-centre, y-centre, width, and height of the bounding box. The box coordinates must be in normalized x y w h format, ranging from 0 to 1. If our original coordinates are in pixels, then we should divide the x-centre and width by the image width, and the y-centre and height by the image height. The class numbers start from 0 (zero-indexed) and represent the different object classes.

2 Brief Introduction of Yolo Model

YOLOv5 (You Only Look Once version 5) is a popular deep learning-based object detection model known for its real-time and efficient performance. It is an evolution of the YOLO series of models, which have significantly contributed to the field of object detection.

The YOLOv5 model builds upon the principles of its predecessors, aiming to improve both accuracy and speed. It follows the single-shot detection approach, meaning that it predicts bounding boxes and class probabilities directly from input images in a single pass, without using complex region proposal networks or multi-stage processing.

One of the key advancements in YOLOv5 is the implementation of a more streamlined architecture that focuses on a balance between accuracy and speed. It introduces a novel backbone network, known as CSPDarknet53, which combines the concepts of Darknet and Cross-Stage Partial Networks to enhance feature extraction capabilities while maintaining computational efficiency.

YOLOv5 also introduces a range of techniques to improve detection performance. It employs a new object detection head, including anchor-free bounding box prediction and focal loss, which helps prioritize hard examples during training. The model also incorporates advanced data augmentation techniques, such as mosaic augmentation and random shapes, to increase robustness and generalization.

Furthermore, YOLOv5 offers flexibility in terms of model size and precision. It provides a series of pre-defined model architectures, such as YOLOv5s, YOLOv5m, YOLOv5l, and YOLOv5x, with varying model depths and computational requirements. This allows users to select the appropriate trade-off between speed and accuracy based on their specific application needs.

YOLOv5 has gained popularity due to its impressive performance on various object detection benchmarks and its ability to run in real-time on both CPUs and GPUs. It has been widely adopted in numerous computer vision applications, including autonomous driving, surveillance systems, robotics, and more.

3 Code explanation and Algorithum

The yolov5 code provided by the repository "urlayatics/yolov5" is an open-source implementation of the YOLOv5 object detection model. This code can be used to train and deploy YOLOv5 models for various object detection tasks.

3.0.1 YoloV5 Model congiguration

The code provides configuration files where we can define the model architecture and settings. These files allow you to specify parameters such as model size (e.g., yolov5s, yolov5m), number of classes, input image size, etc. You can customize these settings based on your specific requirements.

3.0.2 Model Training

Once the dataset and model configuration are prepared, the code allows us to train the YOLOv5 model. During training, the model learns to detect objects by optimizing the network's weights based on the provided dataset. The code includes options for adjusting the training parameters, such as learning rate, batch size, and number of epochs.

3.0.3 Model Evaluation

After training, you can evaluate the performance of the trained model using the validation set. The code provides evaluation metrics such as mean Average Precision (mAP) to assess the model's accuracy.

3.0.4 Model Inference

Once the model is trained and evaluated, we can use it to perform object detection on new images or videos. The code includes functions to load the trained model and apply it to input data. It performs object detection by predicting bounding boxes and class labels for objects in the input images.

3.0.5 Deployment and Export

we can export the trained model for deployment in various formats, including TorchScript and mobile optimized.ptl file These formats enable the model to be used in different environments or integrated into other applications like Android apps.

3.1 Yolo V5 Algorithms

The YOLOv5 model architecture consists of a backbone network (CSPDarknet53), followed by additional layers for detection, classification, and feature extraction. The training process involves iteratively updating the model's parameters using labeled training data and optimization algorithms such as stochastic gradient descent (SGD) or Adam.
Input Receive an image as input for object detection.
Model Initialization Load the YOLOv5s model architecture and initialize the model's parameters.
Image Preprocessing Preprocess the input image to make it compatible with the model. This involves resizing the image to a fixed size, normalizing pixel values, and converting it to the appropriate format.
Forward Pass Pass the preprocessed image through the YOLOv5 model. The image goes through the neural network layers, including convolutional, activation, and pooling layers, to extract meaningful features.
Anchor Box Generation Generate anchor boxes of different sizes and aspect ratios that will be used for object detection at different scales in the image.

Object Detection Prediction For each grid cell in the output feature map, predict bounding box coordinates, objectness scores, and class probabilities. This is done by applying a set of convolutional filters and activation functions to the feature map.
Non-Maximum Suppression (NMS) Apply non-maximum suppression to filter out redundant and overlapping bounding box predictions. This step ensures that only the most relevant and accurate bounding boxes are retained.
Post-processing Convert the predicted bounding box coordinates from the grid cell space to the original image space. Apply thresholding on objectness scores and class probabilities to discard low-confidence detections.
Output Return the final set of bounding boxes, their associated class labels, and confidence scores as the output of the YOLOv5 object detection algorithm.

4 Results and Discussion

Model: YOLOv5s
Number of Epochs: 60
Batch Size: 16
Image Size: 640x640
Learning Rate: 0.00802 (initial)
Optimizer: SGD (Stochastic Gradient Descent)
Dataset: Aerial Maritime
Training Metrics:
Best mAP (mean Average Precision) at IoU (Intersection over Union) threshold 0.5: 0.4566
Best mAP at IoU threshold 0.5:0.95: 0.2398
Best Precision: 0.7886
Best Recall: 0.4014
Losses: Box Loss (train/box loss): 0.0485
Class Loss (train/class loss): 0.0200
Object Loss (train/obj loss): 0.0310
Validation Box Loss (val/box loss): 0.0477
Validation Class Loss (val/class loss): 0.0206
Validation Object Loss (val/obj. loss): 0.0193
I ran only 60 epochs. It is important to note that increasing the number of epochs beyond 60 might lead to further improvements in accuracy, as the model continues to learn and refine its predictions. Additionally, fine-tuning the model on other datasets may enhance its performance on different object detection tasks.

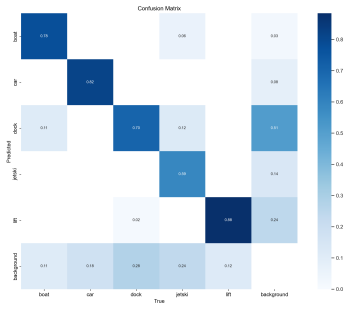

Figure 1: Confusion Metrics

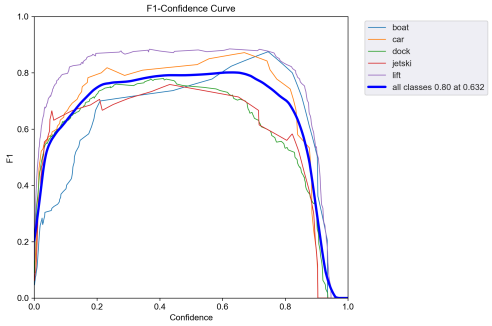

Figure 2: F1-score

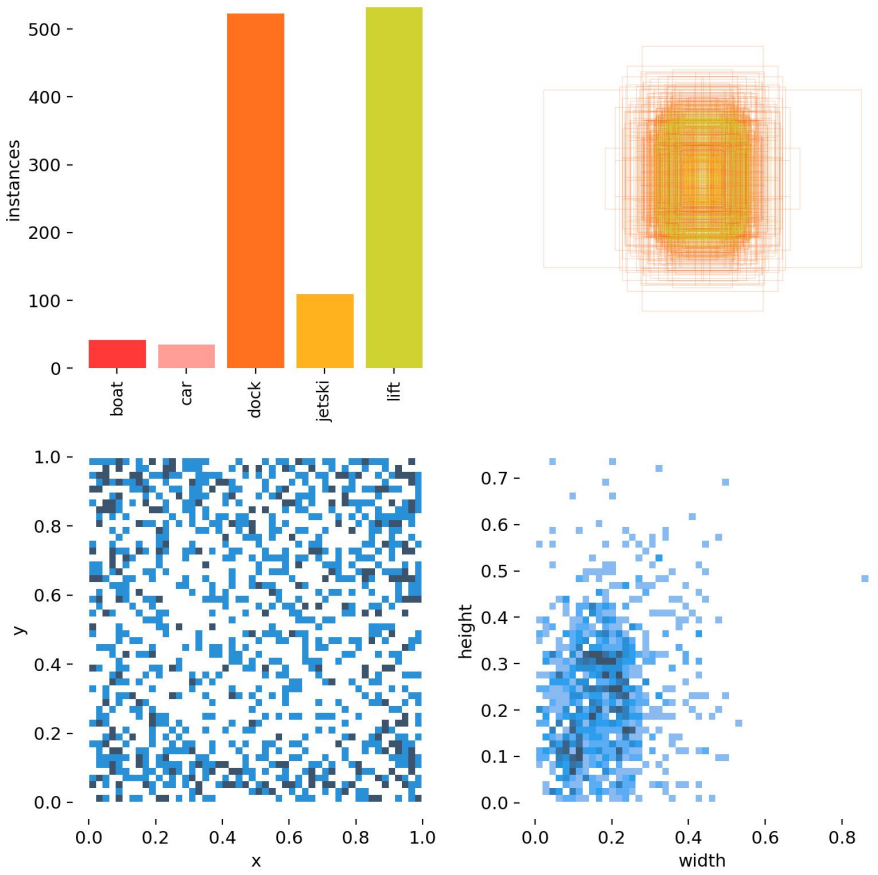

Figure 3: Class instances

Figure 4: Qualitaive Results

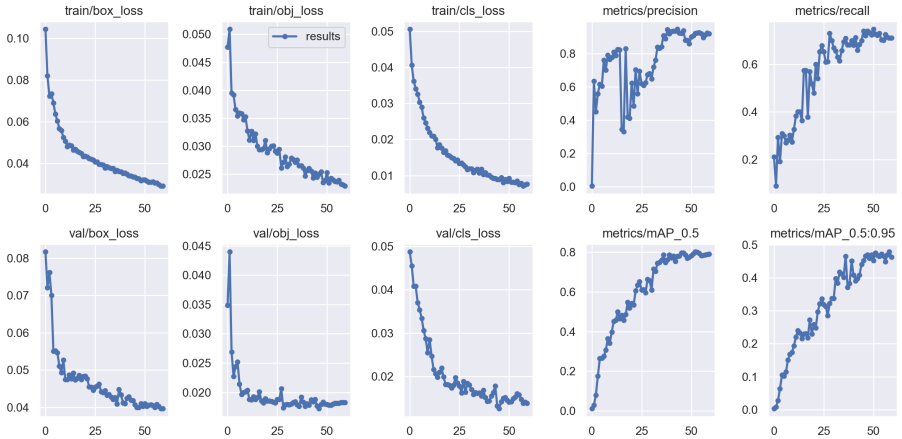

Figure 5: Results

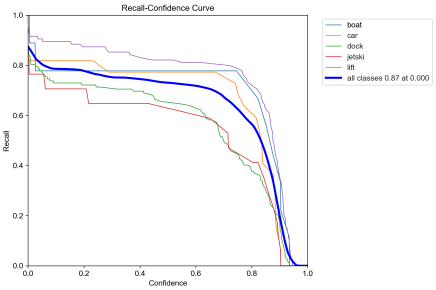

Figure 6: Recall Confidence curve

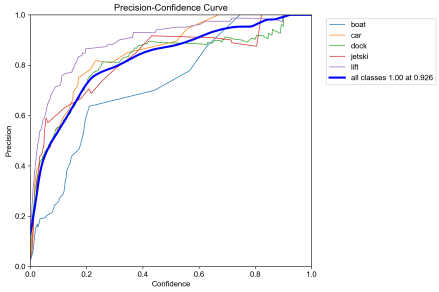

Figure 7: Precision confidence curve

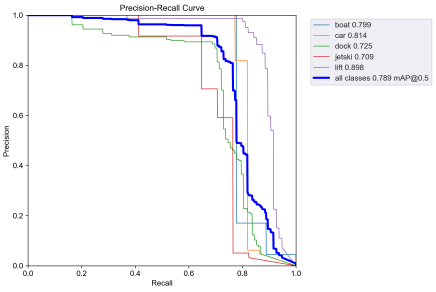

Figure 8: Precession Recall Curve

5 Conclusion

In conclusion, the YOLOv5s model trained on the aerial maritime dataset demonstrates promising results for object detection in aerial images. With 60 epochs of training, the model achieved a mean Average Precision (mAP) of 0.4566 at an Intersection over Union (IoU) threshold of 0.5, indicating its ability to accurately detect and classify objects in the maritime environment.

The model's performance was evaluated on various metrics, including precision, recall, and mAP at different IoU thresholds. The achieved precision of 0.7886 and recall of 0.4014 indicate the model's capability to effectively identify objects of interest. The loss analysis reveals that the model was able to effectively optimize the box, class, and object loss functions during training.

While the model's accuracy can potentially be further improved by increasing the number of epochs, the current results provide a solid foundation. Fine-tuning the model on additional datasets can also enhance its performance on different object detection tasks.

The YOLOv5s model demonstrates practical applications for aerial maritime surveillance, environmental monitoring, and related domains. By accurately detecting objects such as docks, boats, lifts, jetskis, and cars, it can contribute to tasks like monitoring boat traffic, identifying hazards, and assisting in search and rescue operations.

However, it is important to acknowledge the limitations and challenges faced during the experiment. Factors such as occlusion, small object detection, and variations in lighting and weather conditions can impact the model's performance. Further improvements can be explored, such as data augmentation techniques, model architecture modifications, and incorporating additional contextual information to address these challenges.

Overall, the YOLOv5s model exhibits promising results for aerial maritime object detection, showcasing its potential for real-world applications.