computation

MDPI

Article

A Comparative Study of Autonomous Object Detection Algorithms in the Maritime Environment Using a UAV Platform

Emmanuel Vasilopoulos *, Georgios Vosinakis *, Maria Krommyda, Lazaros Karagiannidis, Eleftherios Ouzounoglou and Angelos Amditis

Institute of Communication and Computer Systems (ICCS), 15773 Athens, Greece; maria.krommyda@iccs.gr (M.K.); lkaragiannidis@iccs.gr (L.K.); eleftherios.ouzounoglou@iccs.gr (E.O.); a.amditis@iccs.gr (A.A.)

* Correspondence: emmanouil.vasilopoulos@iccs.gr (E.V.); giorgos.vosinakis@iccs.gr (G.V.)

Abstract: Maritime operations rely heavily on surveillance and require reliable and timely data that can inform decisions and planning. Critical information in such cases includes the exact location of objects in the water, such as vessels, persons, and others. Due to the unique characteristics of the maritime environment, the location of even inert objects changes through time, depending on the weather conditions, water currents, etc. Unmanned aerial vehicles (UAVs) can be used to support maritime operations by providing live video streams and images from the area of operations. Machine learning algorithms can be developed, trained, and used to automatically detect and track objects of specific types and characteristics. EFFECTOR is an EU-funded project, developing an Interoperability Framework for maritime surveillance. Within the project, we developed an embedded system that employs machine learning algorithms, allowing a UAV to autonomously detect objects in the water and keep track of their changing position through time. Using the on-board computation unit of the UAV, we ran and present the results of a series of comparative tests among possible architecture sizes and training datasets for the detection and tracking of objects in the maritime environment. We tested architectures based on their efficiency, accuracy, and speed. A combined solution for training the datasets is suggested, providing optimal efficiency and accuracy.

Keywords: maritime; maritime environment; UAV; UxV; object detection; object tracking; machine learning; situational awareness; computer vision

Citation: Vasilopoulos, E.; Vosinakis, G.; Krommyda, M.; Karagiannidis, L.; Ouzounoglou, E.; Amditis, A. A Comparative Study of Autonomous Object Detection Algorithms in the Maritime Environment Using a UAV Platform. *Computation* **2022**, *10*, 42. https://doi.org/10.3390/computation10030042

Academic Editor: Demos T. Tsahalis

Received: 9 February 2022
Accepted: 14 March 2022
Published: 15 March 2022

1. Introduction

Maritime operations cover a wide variety of scenarios, including search and rescue missions and shipping management. With the increase in marine traffic and evolving maritime climate, shipping management and maritime travel safety have become high-priority issues [1]. The distinct characteristics and ever-changing nature of the maritime environment present unique challenges in surveillance. These include objects drifting and changing location due to the effects of wind and water currents, their position and shape altering due to rolling and pitching caused by waves and wind, while changing weather conditions can radically affect the local situation and outlook [2].

In this elusive environment, timely and precise information is of utmost importance to the success of any mission. Automatic tracking systems and computer vision have been described and utilized in the detection and tracking of several types of objects in the water, from vessels to icebergs, from several platforms [3].

In the maritime field, computer vision and object detection are already being utilized in security and rescue operations [4,5], mainly running on terrestrial modules or modules carried on ships. These data can be incorporated into decision-making algorithms, which can increase the efficiency of the utilized assets [6]. Such systems can be used in border control, as well as search and rescue operations, especially since the two fields frequently

overlap dynamically due to changing conditions in the maritime field. UAVs are being utilized to gather intelligence in a timely manner [7,8]. UAV systems have started to be used in terrestrial surveillance, where the video is streamed to a control station running detection and tracking through computer vision [9].

EFFECTOR [10] is an EU-funded project, aiming to develop an Interoperability Framework and associated data fusion and analytics services for maritime surveillance. The aim of the project is the faster detection of new events, the enhancement of decision-making, and the achievement of a joint understanding and undertaking of a situation by supporting a seamless cooperation between the operating authorities and on-site intervention forces, through a secure and privacy-protected network.

In developing a UAV module for object detection and tracking in the maritime environment, a wide variety of maritime stakeholders, including organizations and institutions involved in all categories of maritime operations, were solicited during the early stages of the project in an effort to identify the most important user requirements. The key requirements identified were:

- **Object detection:** Once an incident has been reported, the UAV will be utilized to rapidly and efficiently scan the area of interest and automatically detect any vessels or humans in the water;
- **Object tracking:** When an object of interest has been detected, the UAV will continuously track its position;
- **Timeliness:** As conditions at sea change rapidly, the system must be able to provide continuous and live detection and tracking of objects;
- **Fast deployment:** The initial stages of a search and rescue are critical, and any system used in this context must be able to be utilized with minimal delay;
- **Interoperability:** Given the abundance of legacy systems used in maritime operations and communications, any new system needs to ensure seamless interoperability with existing systems. Specifically in the EU space, the Common Information Sharing Environment (CISE) [11] is used as the basis for information exchanged among public authorities. Any messages or communications generated by the system should be compatible with the CISE;
- **Independent and self-sufficient operation:** The system must be able to operate independently and be self-sufficient for the duration of the operation;
- **Cost effectiveness:** An on-board UAV surveillance system is inherently cost effective by replacing more expensive and critical airborne assets. By providing timely information of an object's location through a UAV using detection and tracking algorithms, operators will be able to more efficiently utilize all assets involved in the operation (including vessels).

Keeping the above in mind, we developed an autonomous object detection and tracking module based on a neural network trained on specific datasets. A main advantage of our system is that all computation takes place on the on-board AI unit carried by the UAV. The UAV system presented is meant to support two distinct types of maritime operations, shipping management and search and rescue. In such operation, a UAV system would be mainly utilized once an event has been detected to provide live information of the evolving incident.

Deploying a neural network in any system is not a simple process. Multiple hyperparameters, such as the size of the training dataset, the type of architecture, the number of network parameters, and many more, have a huge impact on the final system. Edge devices, installed on UAVs, have limited capabilities in terms of running deep learning models. The model that will be deployed on such a device is required to offer fast forward propagation, with the highest possible precision in detecting and classifying objects inside a given frame.

In the above context, we ran and present in this article a series of comparison tests, using the on-board AI unit of a UAV. Testing focused on comparing different architecture sizes, as well as training datasets for the detection and tracking of objects in the maritime

environment. We compared the architectures based on their efficiency, accuracy, and speed. The main aim was to determine the optimal setup for providing timely data through a live video feed, by minimizing any processing delays, while maintaining low computational demands in order to ensure effective operation using the on-board AI module.

2. System Specifications

2.1. UAV Platform

We ran all tests with the module attached to a specially built octocopter UAV (see Figure 1). It is equipped with a pair of daylight and thermal cameras providing a video stream for the UAV's pilot, enabling extended visual line-of-sight (EVLOS) flight capability. To provide extended capabilities of object detection and tracking, a three-axis stabilized gimbal equipped with powerful RGB and thermal cameras was fit under the UAV and connected to the processing unit for on-board processing. The results of the on-board processing were overlaid on the full-HD video stream and transmitted to the Intelligence Officer's workstation via a high-bandwidth 2.4 GHz radio.

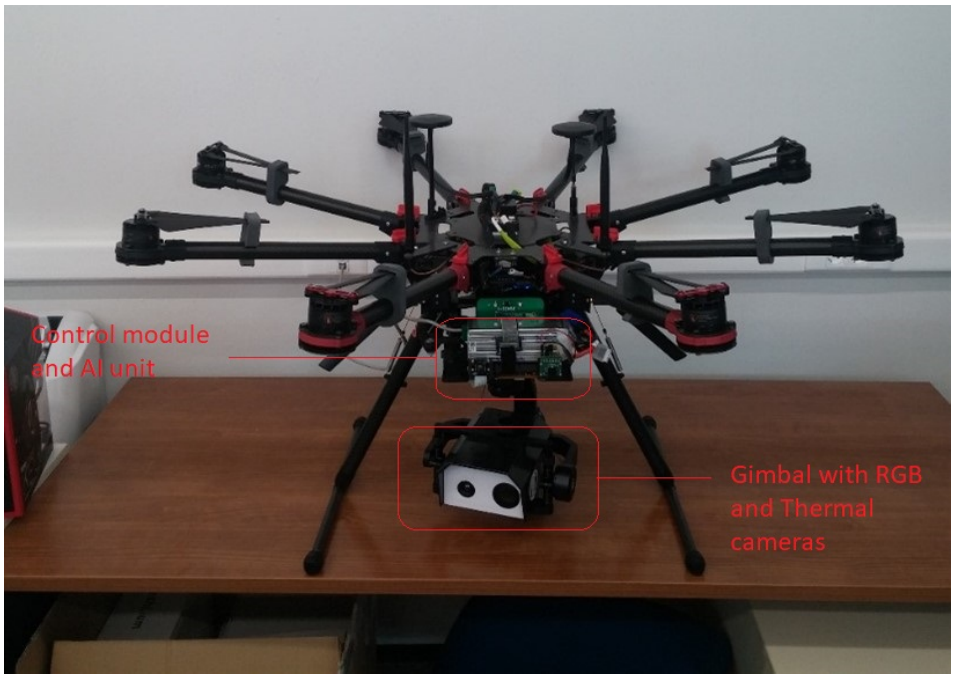

Figure 1. The specially designed octocopter UAV showing the processing module container at its central unit, as well as the gimbal that carries the thermal and RGB cameras suspended underneath.

2.2. Main Computing Module

In our embedded architecture, we opted to run detection and tracking on an on-board processing unit, instead of the Intelligence Officer's ground station where the video feeds are streamed. This was done in line with an edge processing approach to reduce the required bandwidth, improve response time, and avoid delays caused by video encoding and streaming, as well as inevitable interruptions and cuts in the video stream caused by connection issues, obstacles, and weather conditions. Video streaming has high-bandwidth requirements, making it more susceptible to connection problems. In our setup, the UAV transmits detection messages and draws the bounding boxes on-board, removing encoding/decoding and communication latency from the object tracking processing loop. In addition, this approach reduces the load on the Intelligence Officer's workstation, delivering a more scalable solution.

The entire object detection and tracking stack that is described ran on a Jetson AGX Xavier module (see Figure 2). This embedded processing unit is lightweight and has low energy requirements. It compresses the video using dedicated hardware and performs all

necessary computation. The complete technical specifications of the module are presented in Table 1. The unit, due to its small size and power requirements, is limited in terms of processing power (CPU, GPU). The aim was for the computing module to be carried by the UAV and be directly connected to its camera feed, making the system completely autonomous, hence the use of a lightweight, low-power unit. The main aim of the test described was to select the optimal architecture and training set within our restrictions. This module was used in all the comparative tests described below.

Figure 2. The Jetson AGX Xavier module.

Table 1. Technical specifications of the Jetson AGX Xavier module.

GPU	512-core Volta GPU with Tensor Cores
CPU	8-core ARM v8.2 64 bit CPU, 8MB L2 + 4MB L3
Memory	32 GB 256 bit LPDDR4x ǀ 137 GB/s
Storage	32 GB eMMC 5.1
DL accelerator	(2×) NVDLA Engines
Vision accelerator	7-way VLIW Vision Processor
Encoder/decoder	(2×) 4Kp60 ǀ HEVC/(2×) 4Kp60 ǀ 12 bit Support
Size	105 mm × 105 mm × 65 mm
Deployment	Module (Jetson AGX Xavier)

3. System Architecture

The purpose of our implemented architecture was to detect an object in the camera feed, assign it an ID, and then, keep track of its trajectory without altering the ID. Our detection and tracking architecture was composed of two main tools. The stack utilizes YOLOv5 [12], a state-of-the-art object detector. When YOLOv4 was published, it outperformed EfficientDet, the state-of-the-art object detection model up to that time [13]. The comparison described in [13], applied on a V100 GPU, showed that YOLOv4 achieved the same accuracy with EfficientDet at almost double the frames per second (FPS). YOLOv5 performs similarly to YOLOv4. However, YOLOv4 utilizes Darknet, while YOLOv5 utilizes PyTorch, making the latter much easier for training, testing, and deployment [14].

The output of the detector is passed to the DeepSORT tracker. This tracking algorithm's runtime complexity is not perceivably affected by the number of objects tracked since it only processes the position and velocity of tracked objects, while the output of the neural network is independent of the number of objects, greatly reducing the number of parameters that need to be processed. The overall architecture is shown in Figure 3.

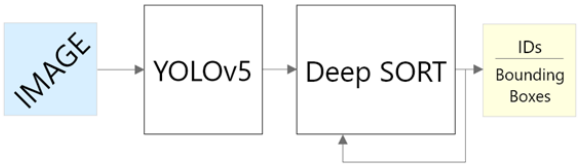

Figure 3. The general architecture showing the distinct components of the system. An image is fed to YOLOv5, and detections are passed to the DeepSORT tracking algorithm. IDs and bounding boxes are retained as the object is tracked.

3.1. Object Detection

YOLOv5 is a high-precision, fast-inference convolutional neural network (CNN) from the YOLO family of object detectors. Its performance is mostly related to its residual connections. It comprises two main components, the backbone and the head, as shown in Figure 4.

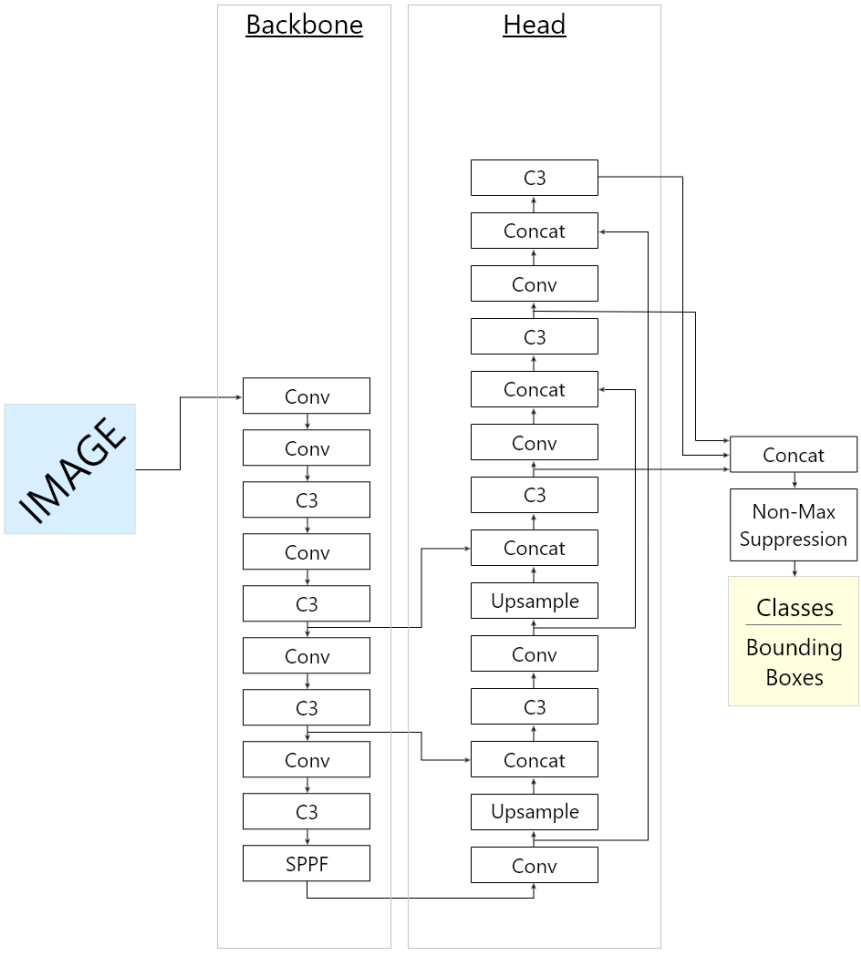

Figure 4. YOLOv5 architecture.

The backbone works as a feature extractor. Starting with simple convolutional layers, the network also uses cross-stage partial [15] layers to further improve efficiency at the intermediate levels. Finally, the spatial pyramid pooling layer (SPP) enhances the receptive field [16]. In other words, the feature produced at the end of this layer contains more information from the input.

The head was also created from convolutional layers and C3 modules, while using features from different stages of the backbone. Other residual connections end up at the

last detection layer, mixing up information from different scales, enabling the network to more easily detect objects of various sizes. The final bounding boxes and their corresponding classes are computed by the non-maximum suppression (NMS), a post-processing algorithm responsible for merging all detections that belong to the same object [17].

The advantage of this architecture over the previous generations is its variation in size. The depth of the network, as well as the width are configurable in tandem. Four options are available: small, medium, large, and extra large. Each size refers to the scale of the model. "Bigger" sizes increase the parameters of the network, leading to longer inference times, but better accuracy. In the context of our use case of an embedded detection unit, we compared the speed of the small and medium versions for use with the Jetson AGX.

3.2. Object Tracking

Multiple object tracking is the problem of keeping track of all objects of interest existing inside a frame. Each object is assigned an ID used to identify each object in the next frames. In our case, within the EFFECTOR project, the user (Intelligence Officer) is able to select which specific detected object the UAV will keep track of among the objects detected. The aim of the tracker in this context is to keep track of objects within a given frame, until the user selects a specific target of interest for the UAV's camera to follow. The difference between detection and tracking is that tracking stores the information of the previous time step and uses it for the current time step.

Simple online and real-time tracking (SORT) is a tracking-by-detection algorithm. It combines Kalman filtering with the Hungarian method [18]. DeepSORT refers to the addition of appearance information to the existing SORT algorithm. To achieve online tracking, detection from the current and the previous frame is presented to the tracker. For every target detected, the tracker keeps a state, comprising its position and velocity.

In this setup, the performance of the tracker is directly improved by the performance of the detector [18]. The runtime of the tracker is highly dependent on the CNN architecture.

4. Experiment Results

4.1. Datasets

4.1.1. SEAGULL Dataset

The SEAGULL dataset [19,20] contains surveillance imagery over the sea. All the content is recorded by a small-sized UAV. Moving cargo ships, small boats, and life rafts are included in the videos, which were captured from different heights and different perspectives. Different cameras were also used, but for the purpose of this paper, we only used videos from RGB cameras and excluded infrared recordings.

The scenarios in the dataset simulate surveillance missions. Thus, life rafts were observed with a length of 3.7 m and a capacity of 20 people. Furthermore, a 27 m-long patrol boat was captured that was either stopped or moving at a speed between 4 kn and 18 kn.

The UAV recording the videos flew between 150 m and 300 m above sea level. The camera's position was set at an angle between 0° and 90° relative to the plane defined by the UAV, as shown in Figure 5.

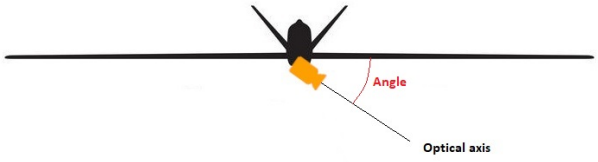

Figure 5. UAV camera position on the UAV showing the angle of the optical axis (direction where the camera is targeted) relative to the plane of the UAV. An angle of 90° between the two means that the camera is pointing directly downwards. (Source: [20]).

4.1.2. COCO Dataset

The COCO dataset [21,22] consists of photos of complex everyday scenes containing objects of 91 different types. It is available under a Creative Commons Attribution 4.0 License. It comes in various versions, and for the purpose of this paper, we used the 2017 release. The training set is made up of 118,000 annotated images and a validation set of 5000 images.

The advantage of this dataset, besides the multiple object types and corresponding instances, is the variation of bounding box sizes. In contrast to SEAGULL, COCO contains objects of numerous scales, from a large airplane that covers almost the whole image, to a tiny tie on a person's suit. The section describing the experiments shows the necessity of this property, in order to avoid outliers and unpredictable occasions where an object is close to the point of recording.

4.1.3. Dataset Management

The reason for combining a general-use object detection dataset (COCO) with a domain-specific dataset (SEAGULL) was to improve the performance of the deployed model. SEAGULL contains a narrow spectrum of bounding box sizes. On the other hand, COCO solves this problem by providing the whole range of bounding box sizes, from the tiniest to one covering the whole image. In addition, the COCO dataset contains images with "sky objects" such as birds.

The camera of the UAV points down from the octocopter at up to 300 m above sea level. While the objects from that point of view will always appear small inside the frame of the camera, there are birds that can fly close to the camera. The neural network will either not detect these birds or will confuse the detection of vessels if those two kinds of "objects" appear simultaneously, since the simple dataset does not contain such data. The necessity of importing COCO to SEAGULL is obvious, considering the expensive equipment of the UAV and the UAV itself, because this can alert its controller and dangerous cases will be avoided in time. Selecting specific objects from COCO more related to a use case might improve the performance even further, but this study is not within the scope of this paper.

The purpose of the experiments was to address two independent matters related to the performance of the deployed model: first, the accuracy and, second, its speed. To achieve this, we split the SEAGULL dataset into a training set and a testing set with the former having 70% and the latter 30% of the dataset. From the COCO dataset, the validation set consisting of 5000 images was used.

4.2. Accuracy

4.2.1. Description and Methodology

Object detection is a difficult task to face in a maritime environment. Sunlight, fog, sea currents, occlusions, etc., are the usual obstacles that lower the performance of detection models. Depending on the use case, an adjustment of the dataset is a prerequisite. In addition, applying supervised learning on a neural network means its predictions will be accurate to similar occasions on which the data have been trained. For example, YOLOv5 will detect only small objects if it is trained on small objects. Small objects can mean either that an object is actually small or that the object is far from the camera's position. If the neural network is trained to detect boats that are away from the camera, it will ignore the boats that are close to the camera, which appear large. To test this hypothesis, three experiments were set up.

Two versions of trained "small" YOLOv5 were exported. The first was trained on the SEAGULL dataset, while the second on a new dataset of COCO and SEAGULL combined. They were both compared to a model pretrained on COCO, to test their performance. We validated all three using a SEAGULL validation set and a combined COCO and SEAGULL dataset, as shown in Figure 6.

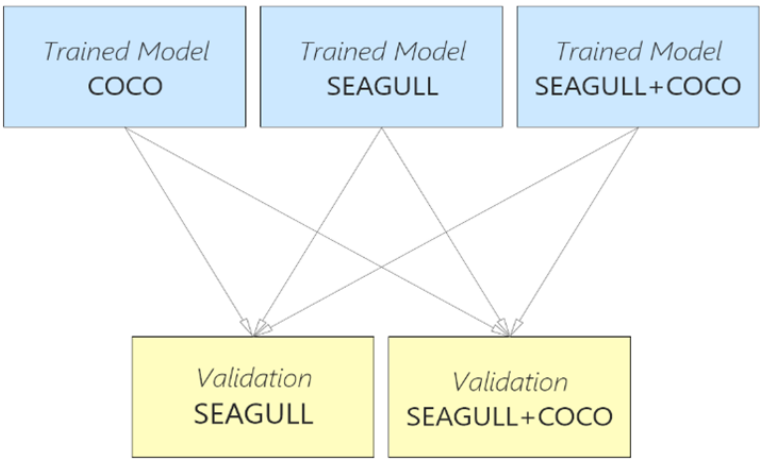

Figure 6. Tested combinations between algorithms and datasets.

All models, except for the pretrained model, were trained for 100 epochs with the basic hyperparameters shown in Table 2. In our hardware described above, the small version of YOLOv5 required 24 h to be trained for 100 epochs with the SEAGULL+COCO dataset. In addition, we noticed when training using the SEAGULL dataset that the mean average precision converged at the 20th epoch, which means that the model will overfit. Thus, we chose to train the models for all experiments for 100 epochs. We opted to train both training experiments for the same amount of time, in order to be able to compare them and derive accurate conclusions.

Table 2. Basic training hyperparameters.

Epochs	100
Batch size	32
Image size	448
Optimizer	SGD
Learning rate	0.01
Momentum	0.937

4.2.2. Metrics

Before demonstrating the results, it is necessary to give a brief explanation of the metrics used to define the performance of the models:

1. **Box loss** measures the difference of the predicted bounding boxes with the ground truth bounding boxes;
2. **Obj loss** measures the difference of the predicted "objectness" with the ground truth "objectness". Objectness is defined as the existence of an object or not in an image;
3. **Cls loss** or **Class loss** measures the error between the predicted class for a detected object and its ground truth class (if class loss is absolute zero, it means the model is trained to detect the objects only and not classify them);
4. **Precision** is the fraction of relevant instances among the retrieved instances;
5. **Recall** is the fraction of relevant instances that were retrieved;
6. **mAP_0.5** is the mean average precision as defined by the VOC dataset [23,24];
7. **mAP_0.5:0.95** is the mean average precision as defined by the COCO dataset.

4.2.3. Experiment A Results—Pretrained Model

We began with a pretrained YOLOv5 that was validated on the SEAGULL dataset and a combination of the SEAGULL and COCO datasets. Since the model was pretrained on the general-purpose COCO dataset, we decided to use its results as a basis for comparison with models trained using the maritime-dedicated SEAGULL dataset and the SEAGULL and COCO combination. It is very important to note that this model was trained with

input image resizing to 640, while our models were trained with input image resizing to 448. This was done in order to increase the detection speed in the UAV's on-board AI unit with its limited processing power.

When validating using the SEAGULL dataset, the values of the metrics appear in Table 3, while a sample of annotated images from the test set are presented in Figure 7 along with the ground truth bounding boxes. Figure 8 shows the lack of predicted bounding boxes on the same images by the trained algorithm. It is evident that the algorithm failed to successfully detect any of the objects.

Table 3. Validation on SEAGULL—values of metrics.

mAP_0.5	0.00002
mAP_0.5:0.95e	0.000005
Box loss	0.16
obj loss	0.013

Figure 7. Examples of annotated images in the test set from SEAGULL—ground truth bounding boxes visible.

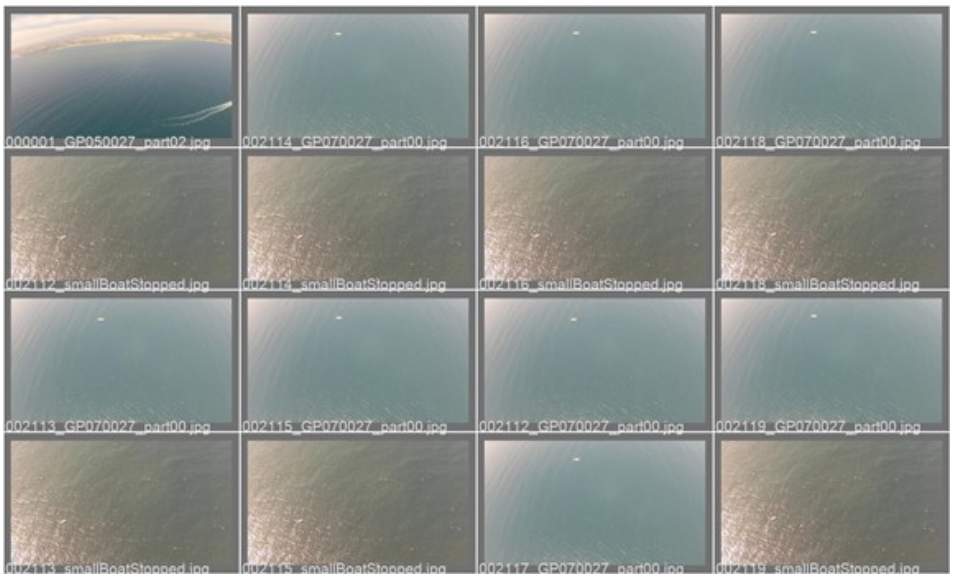

Figure 8. Examples of annotated images in the test set from SEAGULL—the pretrained algorithm failed to predict bounding boxes in all test images.

When using the combined SEAGULL and COCO dataset for validation, the values of the metrics appear in Table 4, while the ground truth bounding boxes on a sample of annotated images from the test set are presented in Figure 9. Once more, no prediction bounded boxes were drawn by the trained algorithm (Figure 10).

Table 4. Validation on SEAGULL and COCO combination—values of metrics.

mAP_0.5	0.0015
mAP_0.5:0.95e	0.0003
Box loss	0.20
obj loss	0.018

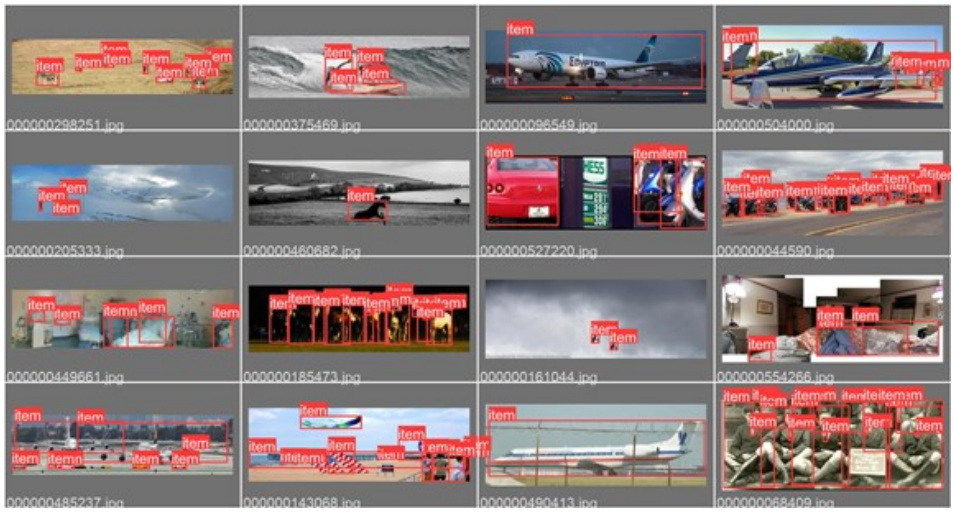

Figure 9. Examples of annotated images in the test set from the COCO and SEAGULL combination—ground truth bounding boxes visible.

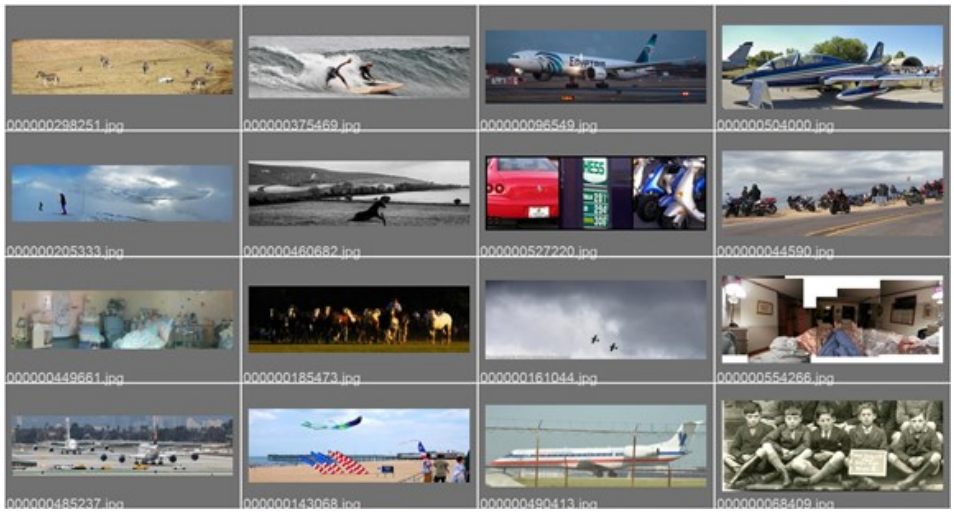

Figure 10. Examples of annotated images in the test set from the COCO and SEAGULL combination—the pretrained algorithm failed to predict bounding boxes in all test images.

Result: It is obvious that the pretrained model did not perform the task of object detection at all. This was mainly due to the input image resizing to 448. Due to our processing power and time limitations, this resizing was a necessary process.

4.2.4. Experiment B Results—Training on the SEAGULL Dataset

This model, as mentioned, was trained on the maritime-surveillance-specific SEAGULL dataset and validated on the SEAGULL dataset and a combination of the SEAGULL and COCO datasets.

Some examples of the frames extracted from videos of SEAGULL are shown in Figures 11 and 12. Both groups of frames display the same frames. Figure 11 shows the ground truth bounding boxes that SEAGULL provides and Figure 12 the predicted bounding boxes by the trained algorithm with the corresponding confidence. These figures present difficulties, such as boats at long distances and very small objects floating on sun glares. In calm waters, the very small objects were detected, which demonstrates the success of the model. The other cases, where the objects were not detected, were predictable because of two main reasons. The first was the low resolution of the input image and the second the very small objects that reside in areas of the frame where the object was not distinguishable from the sun glares, waves, or surrounding environment.

Figure 11. Examples of annotated images from the SEAGULL test set—ground truth bounding boxes visible.

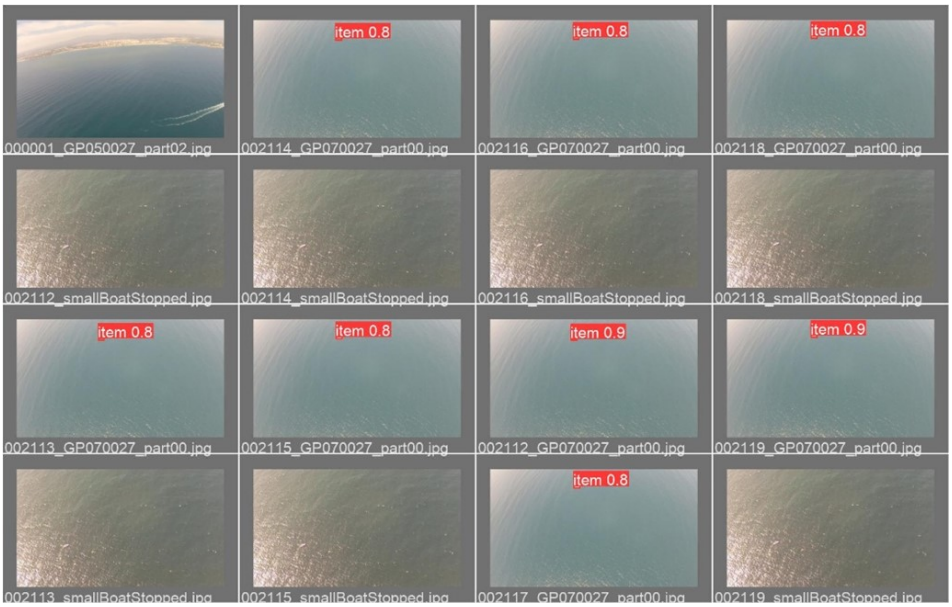

Figure 12. Examples of annotated images from the SEAGULL test set—predicted bounding boxes with the confidence visible as drawn by the algorithm trained on SEAGULL alone.

The progress of training during the 100 epochs is shown in Figures 13 and 14. In Figure 13, in the training losses (first row), class loss was zero since we did not apply object classification. The progress of objectness loss showed that the model detected objects from the start of training, reached a peak for the first epochs, and converged at the 100th epoch. The box loss progression implies that as more epochs passed, the model performed well on the training dataset. In the validation losses (second row), class loss was zero as well. In contrast with the training objectness loss, the validation objectness loss fell very quickly during the first few epochs. However, there was no significant improvement during the next epochs. On the other hand, the more epochs the model was trained for, the worse it performed on the SEAGULL validation dataset. The same applied for the validation box loss.

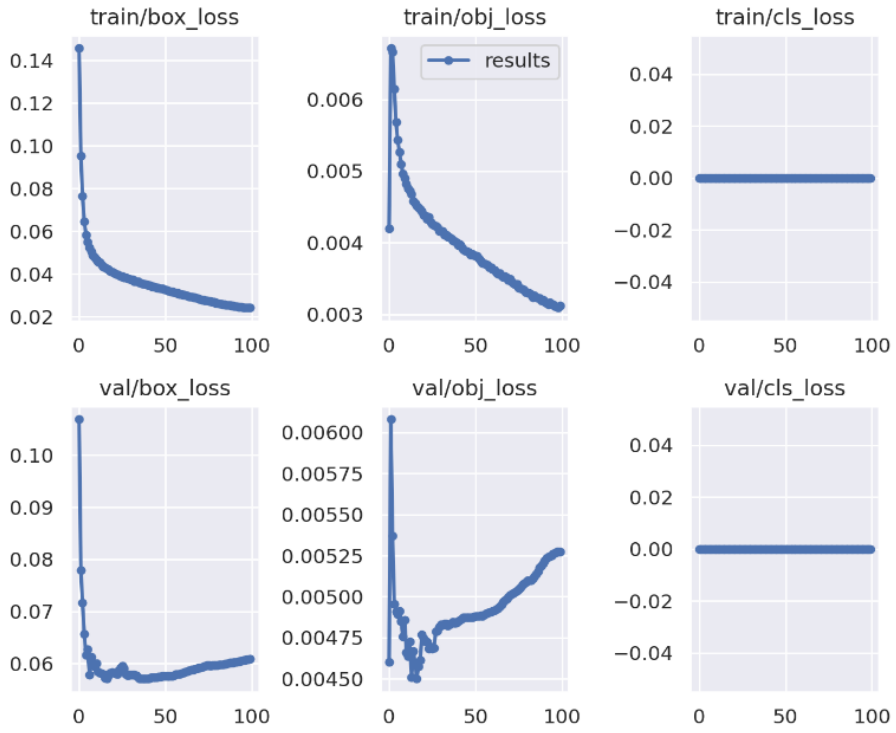

Figure 13. Training on the SEAGULL—progress of metrics during the 100 epochs.

In Figure 14, during the first 15 epochs, all metrics converged on a constant value. What can be derived from the precision and recall graphs is that false positive and false negative detections did not decrease and oscillated around 0.85 and 0.6, respectively, after the 15th epoch. Furthermore, the mean average precisions implied the same conclusion, as well as that the overall performance did not improve.

It is noted that after the 20th epoch, there was no significant progress to the model's performance. The training losses decreased with progressive epochs; however, what matters is the validation losses and most of all the mean average precisions, since the data used in the validation set were data the neural network had not seen.

The optimal values reached for the metrics of interest when validating with the training dataset (SEAGULL) are shown in Table 5.

Table 5. Validation on SEAGULL—optimal values of metrics.

mAP_0.5	0.67
mAP_0.5:0.95e	0.357
Box loss	0.057
obj loss	0.0045

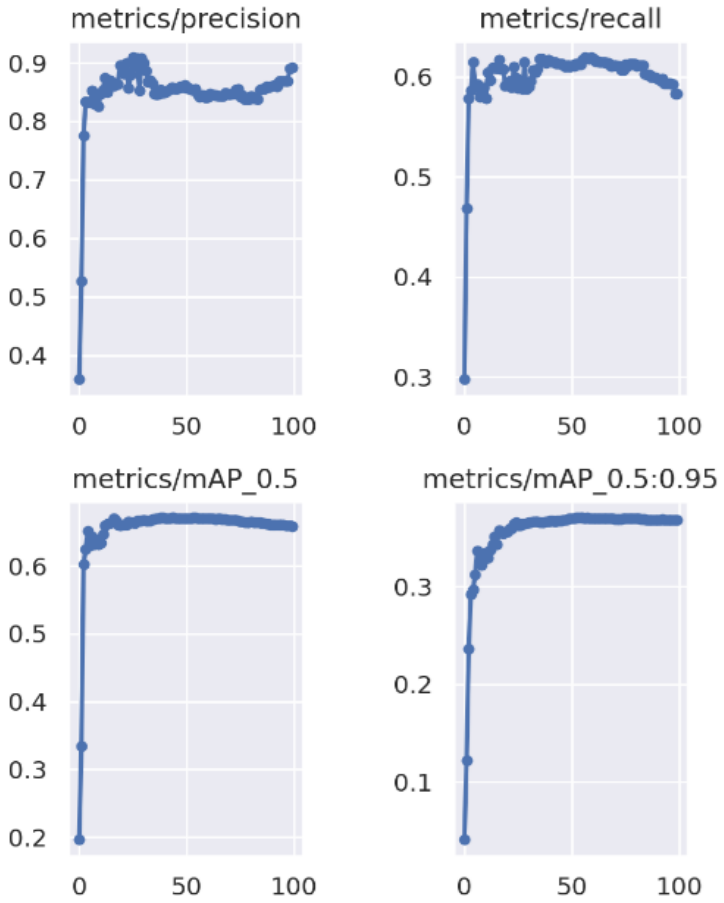

Figure 14. Training on the SEAGULL—progress of metrics during the 100 epochs.

Then, the model was validated on the SEAGULL and COCO combination. The resulting metrics are shown in Table 6.

Table 6. Validation on SEAGULL and COCO combination—values of metrics.

mAP_0.5	0.31471
mAP_0.5:0.95e	0.12733
Box loss	0.055589
obj loss	0.0093003

Result: Comparing the two runs, we see that when validated using the mixed dataset, the mean average precisions were significantly lower, while the validation losses were not affected considerably.

4.2.5. Experiment C Results—Training on the Combined Dataset

The final experiment consisted of a model trained on the mixed SEAGULL and COCO datasets. We validated this using the combined dataset and then the SEAGULL dataset alone as with the previous models.

Some examples of the images included in COCO are displayed in Figures 15 and 16. Both figures display the same group of samples. Figure 15 shows the ground truth bounding boxes with corresponding classes of the objects, and Figure 16 shows the predicted bounding boxes with their classes and the level of confidence. Almost all objects were detected within an image, despite their compact density and the complexity of the image, with minor false negatives and false positives. The confidence of detection was higher when objects were not occluded and distinct.

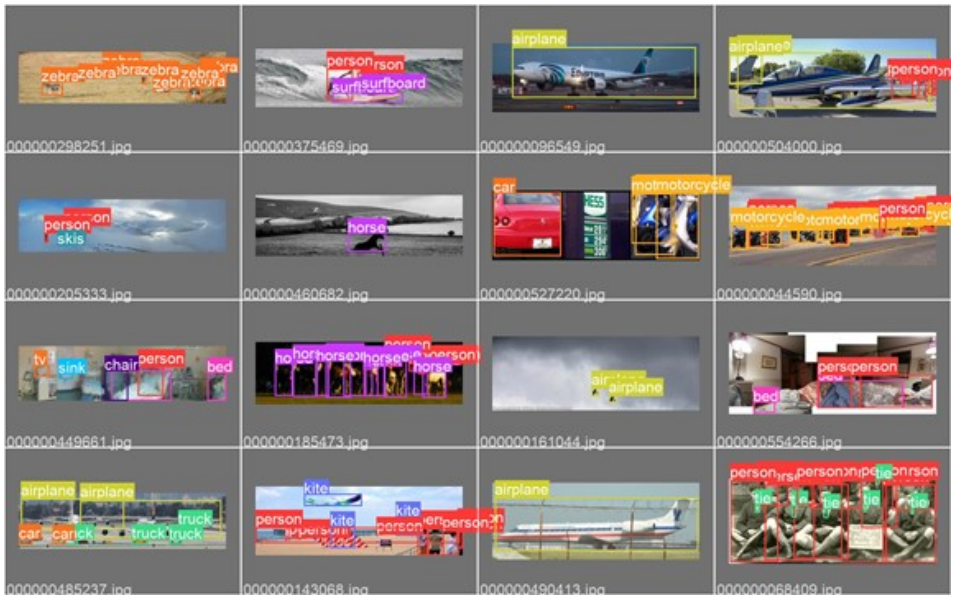

Figure 15. Examples of annotated images from the COCO test set—ground truth bounding boxes and classes visible.

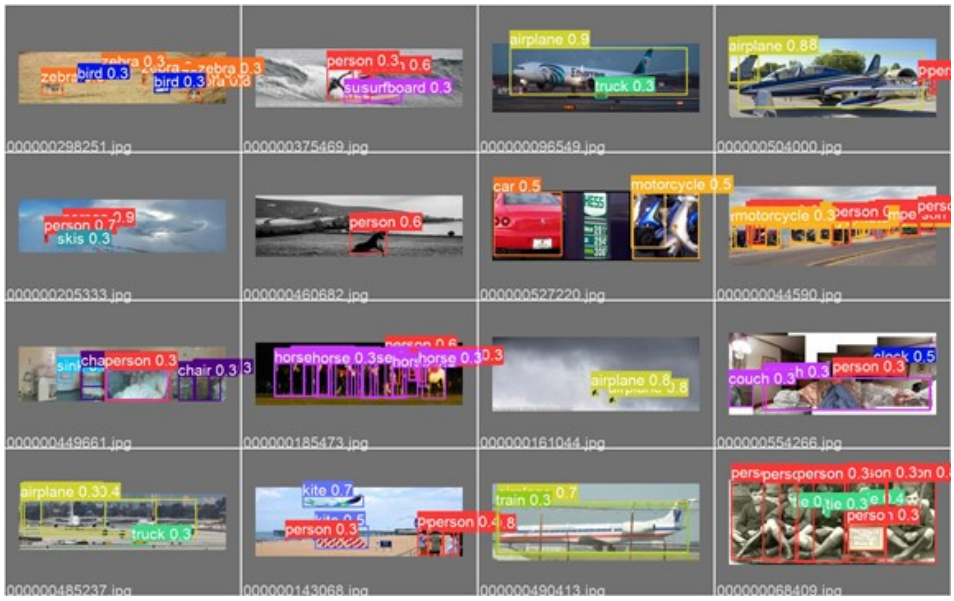

Figure 16. Examples of annotated images in the test set from the COCO and SEAGULL combination—bounding boxes and classes with confidence shown as predicted by the algorithm trained on the combined dataset.

The progress during the 100 epochs of training is shown in Figures 17 and 18. In Figure 17, in the training losses (first row), class loss decreased more as more epochs passed and converged at 0.021. The progress of objectness loss decreased and showed potential for even further reduction beyond the 100th epoch training. The same applied for the box loss. In the validation losses (second row), class loss stabilized around 0.012 after 50 epochs. Objectness loss showed potential for further reduction if we trained the model even more. This was in contrast to the first experiment. From this metric alone, it can be concluded that the model's performance in accuracy can be improved even more. The same applied for the box loss. Neither converged on a value after 100 epochs and had a descending trajectory.

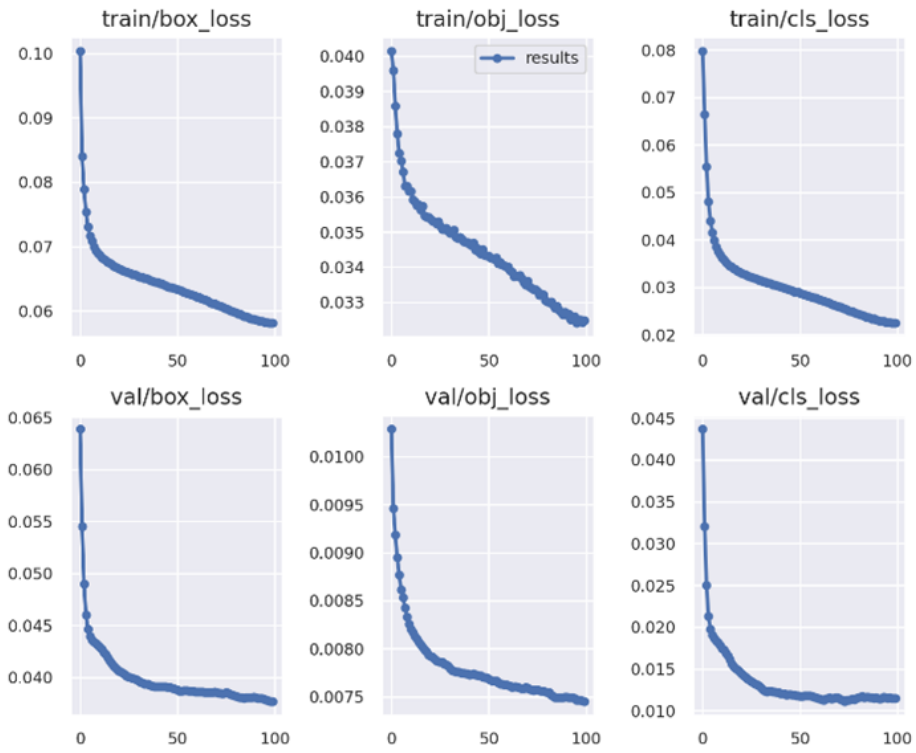

Figure 17. Training on the SEAGULL and COCO combination—progress of metrics during the 100 epochs.

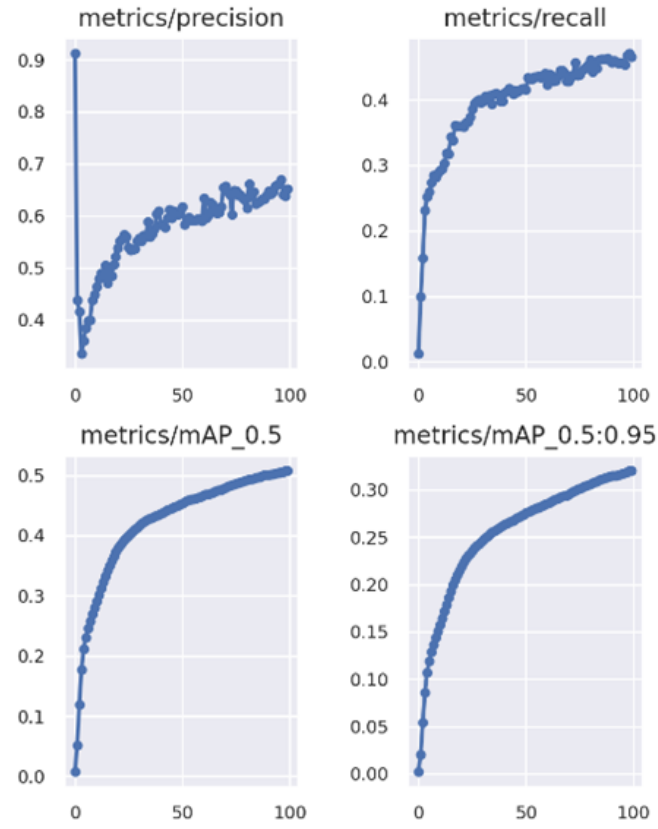

Figure 18. Training on the SEAGULL and COCO combination—progress of metrics during the 100 epochs.

The same conclusion as Figure 17 can be derived after studying the metrics of Figure 18. The ascending course of precision and recall, as well as the mean average precisions showed the setup's potential for a better final model with more epochs of training. None converged on a single value.

The initial value of precision, which was 0.9, is rational considering recall, which had a value of zero at the first epoch. A precision of 0.9 means that the model predicted/detected with very few false positives, while a recall of zero means the model predicted/detected with many false negatives.

The optimal values for the metrics of interest are shown in Table 7.

Table 7. Validation on SEAGULL and COCO combination—values of metrics.

mAP_0.5	0.50
mAP_0.5:0.95e	0.32
Box loss	0.037
obj loss	0.007

The model trained on the combined dataset was then validated on the SEAGULL validation set alone. The resulting metrics are shown in Table 8.

Table 8. Validation on SEAGULL—values of metrics.

mAP_0.5	0.66
mAP_0.5:0.95e	0.337
Box loss	0.031
obj loss	0.0034

Result: It is obvious that further training would result in outperforming the model from Experiment B (Section 4.2.4). The difference in the respective metrics were minuscule, concluding the superiority of the second model trained on the combination dataset.

This model trained on SEAGULL and COCO was much more generalized than the model trained on SEAGULL only (Section 4.2.4). The former was able to detect objects close to the position of the camera that appeared larger inside the frame. The latter, on the other hand, was unable to make the same predictions, since it was trained to detect objects that appear at a certain distance from the camera. This means that a certain range of object sizes only was detectable by the model.

Comparing the two models, in Figures 19 and 20, Graphs (a) represent the number of training items in the dataset, Graphs (b) the anchor boxes used for the final bounding boxes, i.e., the detections, Graphs (c) the frequency of the central positions of the objects of the dataset, and Graph (d) the size of the bounding boxes as a percentage of the images' width and height for the SEAGULL- and SEAGULL+COCO-trained models, respectively.

The SEAGULL dataset contains data with bounding boxes with a maximum height of 10% and a maximum width of 15% of the respective dimensions of the image. On the contrary, the SEAGULL+COCO dataset is uniform in relation to the anchor boxes, central positions, and bounding box sizes. This means that the model was trained on a larger variety of objects and sizes, respectively, allowing for detections on more unpredictable cases where the object to be detected appears larger due to the proximity to the camera.

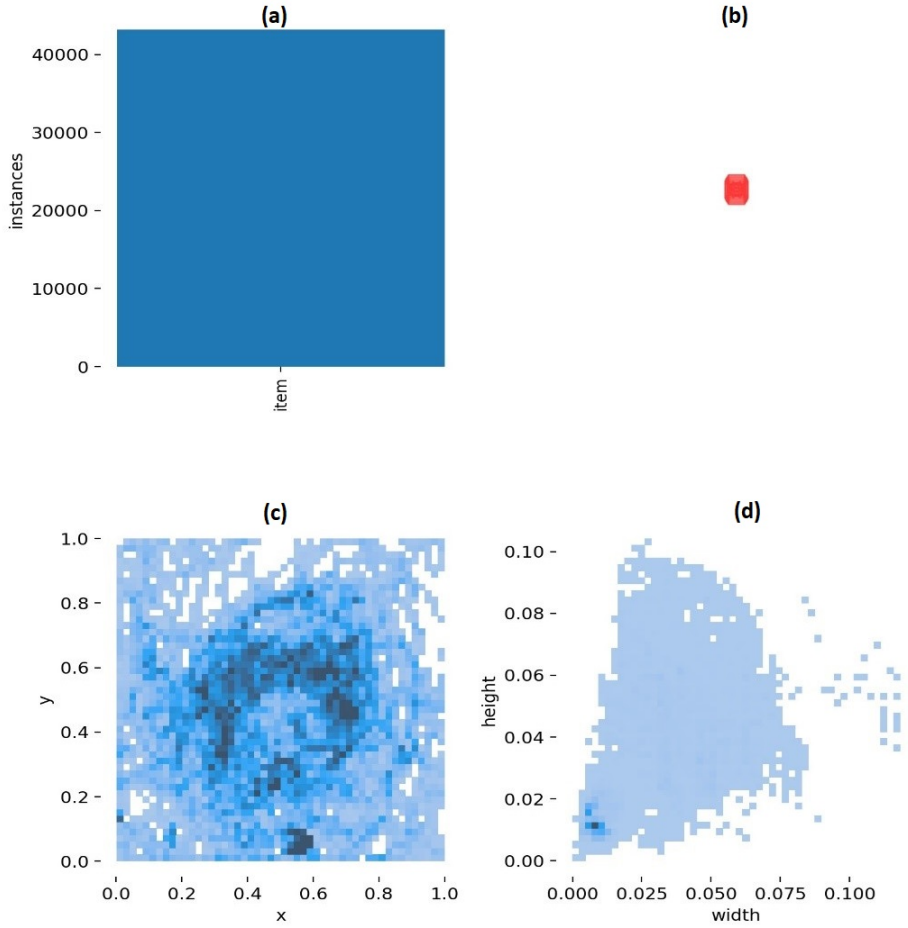

Figure 19. Model trained on SEAGULL—(**a**) number of training items in the dataset, (**b**) anchor boxes used for the final bounding boxes, i.e., the detections, (**c**) frequency of the central positions of the objects of the dataset, (**d**) size of the bounding boxes as a percentage of the images' width and height for the SEAGULL- and SEAGULL+COCO-trained models, respectively.

4.3. Speed

4.3.1. Description and Methodology

All deep learning models may achieve the task at hand; however, some applications require their output in real-time. In our case, the deployed hardware on the UAV, i.e., the Jetson AGX module, may be optimized for running deep learning models, but its limitations are obvious.

The next experiments compare the trade-off between accuracy and speed. YOLOv5, as stated, comes in various sizes. The small version's weights take a space of 20 MB, while the medium version's require 70 MB. The difference in memory allocation is significant, and we compared the accuracy and speed for each. For the purpose of this experiment, we chose the best-performing dataset from the previous set of tests, i.e., the SEAGULL and COCO combination.

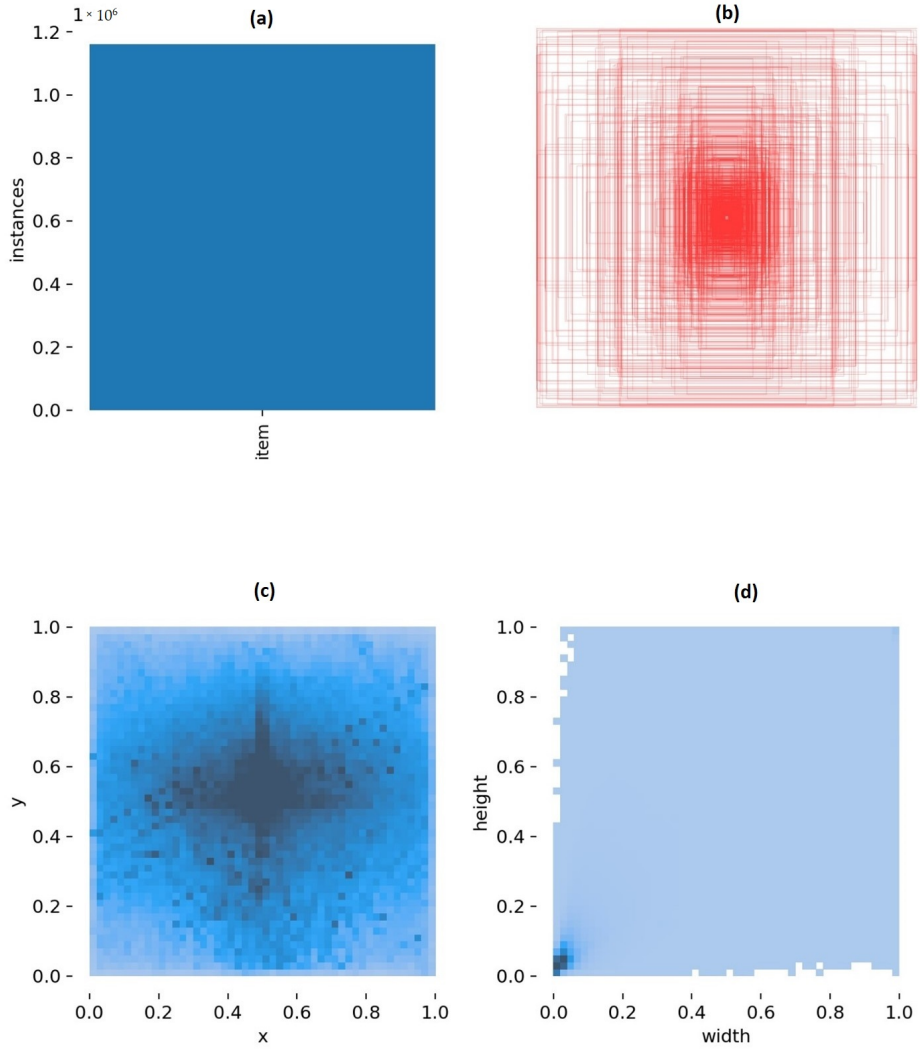

Figure 20. Model trained on SEAGULL and COCO—(**a**) number of training items in the dataset, (**b**) anchor boxes used for the final bounding boxes, i.e., the detections, (**c**) frequency of the central positions of the objects of the dataset, (**d**) size of the bounding boxes as a percentage of the images' width and height for the SEAGULL- and SEAGULL+COCO-trained models, respectively.

4.3.2. Results

Using the small version of YOLOv5, the forward pass of one frame on the neural networks lasted for around 70 ms. The medium version took around 120 ms to process one frame. The performance regarding the accuracy for the two versions is shown in Table 9.

Table 9. Comparison of metrics between the small and medium versions of YOLOv5.

Metrics	Small YOLOv5	Medium YOLOv5
mAP_0.5	0.50	0.57
mAP_0.5:0.95e	0.32	0.38
Box loss	0.037	0.034
obj loss	0.007	0.007

The difference in accuracy of each model was not minor; however, the difference in inference time was significant. As our stack used DeepSORT for object tracking, which requires the arrival of detections as quickly as possible, we opted for the small version of YOLOv5 for deployment.

5. Conclusions

Our sequential detection and tracking architecture consisted of the YOLOv5 object detector and the DeepSORT object tracker. Our use case was based on an on-board, autonomous AI unit, carried by a UAV and directly processing the live video stream from the UAV's cameras. The main advantage of this embedded architecture is that detection and tracking do not depend on the quality of the video stream since all detection and tracking happens on-board the UAV, while the reduced load on the ground units due to edge processing offers a more scalable solution. Given the restrictions in computational power of the on-board Jetson AGX Xavier unit, as well as the fact that DeepSORT utilizes the result of object detection in tracking, speed is critical in the object detection stage, as delays in detection will affect the efficiency of tracking. Comparing the accuracy of the medium and small YOLOv5 versions, we opted for the small version, which enhanced the speed without critically compromising accuracy.

In training the YOLOv5 detector, we used the maritime-surveillance-dedicated SEAGULL and the COCO dataset. The model trained using SEAGULL performed well when validated using images from the same dataset, but a model trained on a combination of both datasets clearly outperformed it with further training. The model trained on the combination dataset was thus superior for our purposes.

In terms of speed, the medium YOLOv5 architecture outperformed the small architecture, as expected, but the small architecture was much faster, so much more appropriate for our real-time detection and tracking setup.

Within the context of the EFFECTOR project, the system will be tested further, including a series of thorough trials and validation of the on-board detection and tracking module in field conditions.

Author Contributions: Conceptualization, E.V., G.V. and M.K.; methodology, E.V. and G.V.; software, E.V.; formal analysis, E.V.; data curation, E.V.; writing—original draft preparation, E.V. and G.V.; supervision, M.K., L.K., E.O. and A.A.; project administration, L.K., E.O. and A.A.; funding acquisition, A.A. All authors have read and agreed to the published version of the manuscript.

Funding: This work is part of the EFFECTOR project. EFFECTOR has received funding from the European Union's Horizon 2020 Research and Innovation Programme under Grant Agreement No. 371883374. The content reflects only the authors' views.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

UAV	Unmanned aerial vehicle
CNN	Convolutional neural network
CSP	Cross-stage partial
SPP	Spatial pyramid pooling
SORT	Simple online and real-time tracking
CISE	Common Information Sharing Environment
EVLOS	Extended visual line-of-sight

References

1. Rubin, A.; Eiran, E. Regional maritime security in the eastern Mediterranean: Expectations and reality. *Int. Aff.* **2019**, *95*, 979–997. [CrossRef]
2. Breivik, Ø.; Allen, A.A.; Maisondieu, C.; Roth, J.C. Wind-induced drift of objects at sea: The leeway field method. *Appl. Ocean. Res.* **2011**, *33*, 100–109. [CrossRef]
3. Silva, T.A.M.; Bigg, G.R. Computer-based identification and tracking of Antarctic icebergs in SAR images. *Remote Sens. Environ.* **2005**, *94*, 287–297. [CrossRef]
4. Pelot, R.; Akbari, A.; Li, L. Vessel Location Modeling for Maritime Search and Rescue. In *Applications of Location Analysis*; International Series in Operations Research & Management Science; Eiselt, H., Marianov, V., Eds.; Springer: Cham, Switzerland, 2015; Volume 232. [CrossRef]

5. Bürkle, A.; Essendorfer, B. Maritime surveillance with integrated systems. In Proceedings of the 2010 International WaterSide Security Conference, Carrara, Italy, 3–5 November 2010; pp. 1–8. [CrossRef]
6. Agbissoh OTOTE, D.; Li, B.; Ai, B.; Gao, S.; Xu, J.; Chen, X.; Lv, G. A Decision-Making Algorithm for Maritime Search and Rescue Plan. *Sustainability* **2019**, *11*, 2084. [CrossRef]
7. Klein, N. Maritime autonomous vehicles and international laws on boat migration: Lessons from the use of drones in the Mediterranean. *Mar. Policy* **2021**, *127*, 104447. [CrossRef]
8. Bauk, S.; Kapidani, N.; Sousa, L.; Lukšić, Ž; Spuža, A. Advantages and disadvantages of some unmanned aerial vehicles deployed in maritime surveillance. In *Maritime Transport VIII: Proceedings of the 8th International Conference on Maritime Transport: Technology, Innovation and Research: Maritime Transport'20*; Martínez, F.X., Castells, M., Martín, M., Puente, J.M., Eds.; Departament de Ciència i Enginyeria Nàutiques, Universitat Politècnica de Catalunya: Barcelona, Spain, 2020; pp. 91–102, ISBN 978-84-9880-827-8. Available online: http://hdl.handle.net/2117/329709 (accessed on 8 February 2022).
9. Maltezos, E.; Douklias, A.; Dadoukis, A.; Misichroni, F.; Karagiannidis, L.; Antonopoulos, M.; Voulgary, K.; Ouzounoglou, E.; Amditis, A. The INUS Platform: A Modular Solution for Object Detection and Tracking from UAVs and Terrestrial Surveillance Assets. *Computation* **2021**, *9*, 12. [CrossRef]
10. EFFECTOR Homepage. Available online: https://www.effector-project.eu/ (accessed on 18 January 2022).
11. CISE Homepage. Available online: https://ec.europa.eu/oceans-and-fisheries/ocean/blue-economy/other-sectors/common-information-sharing-environment-cise_en (accessed on 18 January 2022).
12. YOLOv5 Repository. Available online: https://github.com/ultralytics/yolov5 (accessed on 18 January 2022).
13. Tan, M.; Pang, R.; Le, Q.V. Efficientdet: Scalable and efficient object detection. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Seattle, WA, USA, 14–19 June 2020; pp. 10781–10790.
14. Nepal, U.; Eslamiat, H. Comparing YOLOv3, YOLOv4 and YOLOv5 for Autonomous Landing Spot Detection in Faulty UAVs. *Sensors* **2022**, *22*, 464. [CrossRef] [PubMed]
15. Wang, C.-Y.; Liao, H.-Y.M.; Wu, Y.-H.; Chen, P.-Y.; Hsieh, J.-W.; Yeh, I.-H. CSPNet: A New Backbone that can Enhance Learning Capability of CNN. In Proceedings of the 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), Seattle, WA, USA, 14–19 June 2020; pp. 1571–1580. [CrossRef]
16. Bochkovskiy, A.; Wang, C.-Y.; Liao, H. YOLOv4: Optimal Speed and Accuracy of Object Detection. *arXiv* **2020**, arXiv:2004.10934.
17. Hosang, J.; Benenson, R.; Schiele, B. Learning Non-maximum Suppression. In Proceedings of the 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, USA, 21–26 July 2017; pp. 6469–6477. [CrossRef]
18. Bewley, A.; Ge, Z.; Ott, L.; Ramos, F.; Upcroft, B. Simple online and realtime tracking. In Proceedings of the 2016 IEEE International Conference on Image Processing (ICIP), Phoenix, AZ, USA, 25–28 September 2016; pp. 3464–3468. [CrossRef]
19. Seagull Dataset. Available online: https://vislab.isr.tecnico.ulisboa.pt/seagull-dataset/ (accessed on 18 January 2022).
20. Ribeiro, R.; Cruz, G.; Matos, J.; Bernardino, A. A Data Set for Airborne Maritime Surveillance Environments. *IEEE Trans. Circuits Syst. Video Technol.* 2019, 29, 2720–2732. [CrossRef]
21. COCO Homepage. Available online: https://cocodataset.org/#home (accessed on 18 January 2022).
22. Lin, T.-Y.; Maire, M.; Belongie, S.; Hays, J.; Perona, P.; Ramanan, D.; Dollár, P.; Zitnick, C.L. Microsoft coco: Common objects in context. In *Proceedings of the European Conference on Computer Vision*; Springer: Cham, Switzerlands, 2014; pp. 740–755.
23. Everingham, M.; Eslami, S.M.; Gool, L.V.; Williams, C.K.I.; Winn, J.; Zisserman, A. The pascal visual object classes challenge: A retrospective. *Int. J. Comput. Vis.* **2015**, *111*, 98–136. [CrossRef]
24. VOC Homepage. Available online: http://host.robots.ox.ac.uk/pascal/VOC/ (accessed on 18 January 2022).