Volume XXVII 2024
ISSUE no.2
MBNA Publishing House Constanta 2024

Scientific Bulletin of Naval Academy

SBNA PAPER • OPEN ACCESS

Making Naval Operations Safer and More Secure Using AI Algorithms

To cite this article: V. Dobref, P. Burlacu, E.G. Robe-Voinea, A. F. Ionescu and V. Mocanu, Scientific Bulletin of Naval Academy, Vol. XXVII 2024, pg. 148-156.

Submitted: 23.04.2024
Revised: 01.10.2024
Accepted: 16.10.2024

Available online at www.anmb.ro

ISSN: 2392-8956; ISSN-L: 1454-864X

doi: 10.21279/1454-864X-24-I2-017

Making Naval Operations Safer and More Secure Using AI Algorithms

V. Dobref[1], P. Burlacu[2], E.G. Robe-Voinea[3], A. F. Ionescu[4], V. Mocanu[5]

[1]Professor PhD eng, Naval Academy "Mircea cel Bătrân", Constanța, RO
[2]Assoc. Prof. PhD eng, Naval Academy "Mircea cel Bătrân", Constanța, RO
[3] Senior Lect. PhD eng, Naval Academy "Mircea cel Bătrân", Constanța, RO
[4] Senior Lect., PhD, Ovidius University of Constanța, Constanța, RO
[5] Senior Lect. PhD eng, Naval Academy "Mircea cel Bătrân", Constanța, RO

E-mail: elena.robe@anmb.ro

Abstract. Nowadays, AI concepts have become almost indispensable for scientific research, with an exponential growth in their capabilities in various fields. The recent challenges generated by the conflict in the Black Sea Basin have required finding solutions for the search and identification of drifting sea mines, which represent a potential danger to maritime vessels and offshore engineering infrastructure. Consequently, this work reports on the development, up to an intermediate stage, of an autonomous system based on aerial drones (UAV) and surface drones (USV) using AI algorithms totally oriented towards increasing military capabilities.
Keywords: artificial intelligence, machine learning, data augmentation, YOLO algorithm, object detection, prediction, autonomous system.

1. Introduction

The recent challenges generated by the conflict in the Black Sea Basin require finding solutions for the search and identification of drifting sea mines. In this paper, we present an overview of the development, up to an intermediate stage, of an autonomous system based on aerial drones (unmanned aerial vehicles - UAV) and surface drones (unmanned surface vehicles - USV) that uses the YOLO object detection algorithm to detect mines.

The remainder of the paper is structured as follows. Section 2 frames object detection as a machine learning problem. Section 3 highlights several common issues that are relevant for machine learning in general and object detection in particular. In Section 4, we offer a brief description of YOLO as an object detection approach. In Section 5 we describe one of our in-house projects, also suggesting how part of the previously presented notions could be implemented.

2. Object detection in the machine learning context

We will start by framing the object detection techniques used in the broader context of machine learning (ML). There are three major ML paradigms: supervised learning, unsupervised learning, and reinforcement learning. The main difference between supervised and unsupervised learning consists of the usage of ground truth values (i.e., labels/annotations assigned by human decision makers to each instance of the input data): in unsupervised learning, no such labels are provided by humans. This means that the algorithm should discover patterns in the data on its own. As for the reinforcement learning paradigm, it is a different, agent-based approach, where human input includes the goal that the agent should pursue, the environment in which the agent can move, and a reward function for the interaction of the agent with the environment. Then, the agent explores the environment, attempting to optimize its cumulative reward.

Note that object detection and classification regularly fall into the supervised (or sometimes into a semi-supervised) learning category [1], [2], [3], hence the focus of the present paper. As a supervised learning task, object detection typically requires annotating samples.

A very important part of supervised learning is model training. Training generally aims to optimize two main objectives:

1. *speed* (usually measured in terms of time)
2. *accuracy* (measured using various performance metrics depending on the nature of the problem and the model used, reflecting various definitions of the distance between model predictions and actual target values).

Of note, the two main objectives are often conflicting and thus hard to simultaneously optimize, forcing researchers into accepting trade-off solutions (accuracy at the expense of speed or vice versa).

Training an ML model typically involves the steps illustrated in Figure 1. First, we need to collect data - as many samples as possible, under diverse conditions. Second, we need to label/annotate the data, providing the "ground truth" values. Third, we should split this labeled dataset into training data and the so-called "unseen" data used for validation and testing. Training data is the part of the dataset for which the model is provided the corresponding labels. "Unseen" data is the part of the dataset for which labels are not provided to the model. This is the testing data and, optionally, the validation data. Validation data is unseen data used during training (e.g., to detect overfitting, which will be briefly covered in a later subsection). Finally, the actual training step can start from random parameter values or fine-tune parameters of a similar, but more general model which was *pretrained* (if such a model is available). The latter approach is called transfer learning.

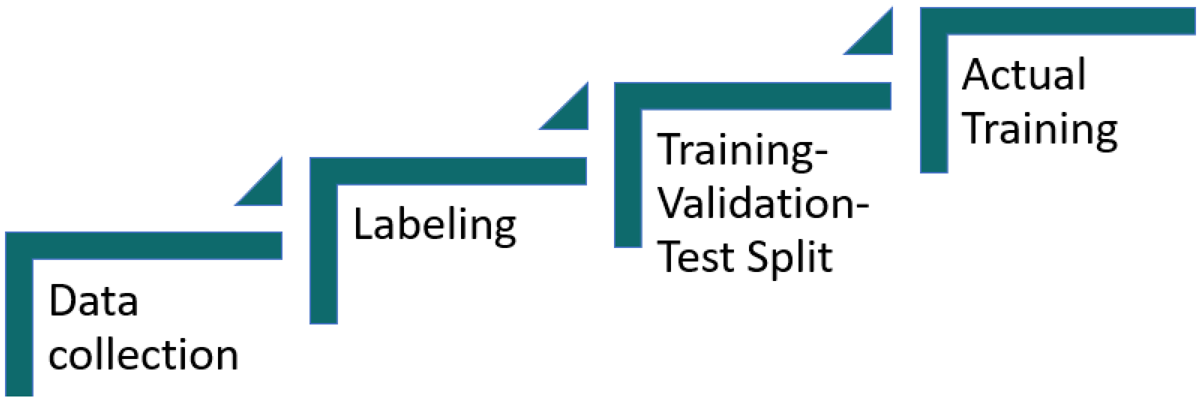

Figure 1. A typical training flow

3. Training issues and solutions

Common training issues include two opposite problems, which are *underfitting* (poor performance on training data) and *overfitting* (high performance on training data, but poor performance on unseen data) [4]. Roughly speaking, these are caused by a mismatch between dataset size (i.e., number of training samples) and model complexity (i.e., number of parameters to be learned).

Typical causes and solutions for the two issues are summarized in Table 1.

Table 1. Underfitting versus Overfitting in ML Model Training

	Underfitting	**Overfitting**
Causes	• Large dataset • Simple model	• Small dataset • Complex model
Solutions	• Increase duration of training • Increase model complexity	• Data augmentation • Early stopping, dropout, and other regularization techniques, normalization techniques

While underfitting can be easy to detect by only looking at the model's performance on the training data (if the accuracy is low, we need to train more or to increase model complexity), overfitting is more difficult to detect because we also need validation data during training. Overfitting may occur because the dataset is too small for the complexity of the model. We can modify the duration of training or restrict the parameters of the model.

A common workaround for overfitting is data augmentation, which involves creating synthetic training instances based on the existing training instances. Such an approach may often prove useful in the military context, where scarcity of data seems to be a common problem. Thus, we have a high risk of overfitting because of small datasets, or datasets that fail to capture certain aspects which may occur naturally in input data (e.g., no images of sea mines under rain conditions, which may occur in practice). Effects imitating such conditions may sometimes be synthetically generated. Finally, the augmented data is added to the existing dataset in order to increase its size.

For images, popular data augmentation operations include affine transforms, filtering transforms, and "functional" transforms. There are multiple Python libraries that provide convenient functions for image augmentation. One of the most recently developed is the Albumentations library [5], an open-source library implemented as a wrapper for multiple existing augmentation libraries, providing a simplified interface to them. The augmentations presented in Figures 2-4 were generated using the Albumentations library.

Affine geometric transforms include flipping, rotations, scaling, shearing, and translations. Figure 2 illustrates several such transforms applied to an image containing a sea mine. Of note, we can compose transforms, as you can see in the lower right image, in which three geometric transforms were successively applied to the original image.

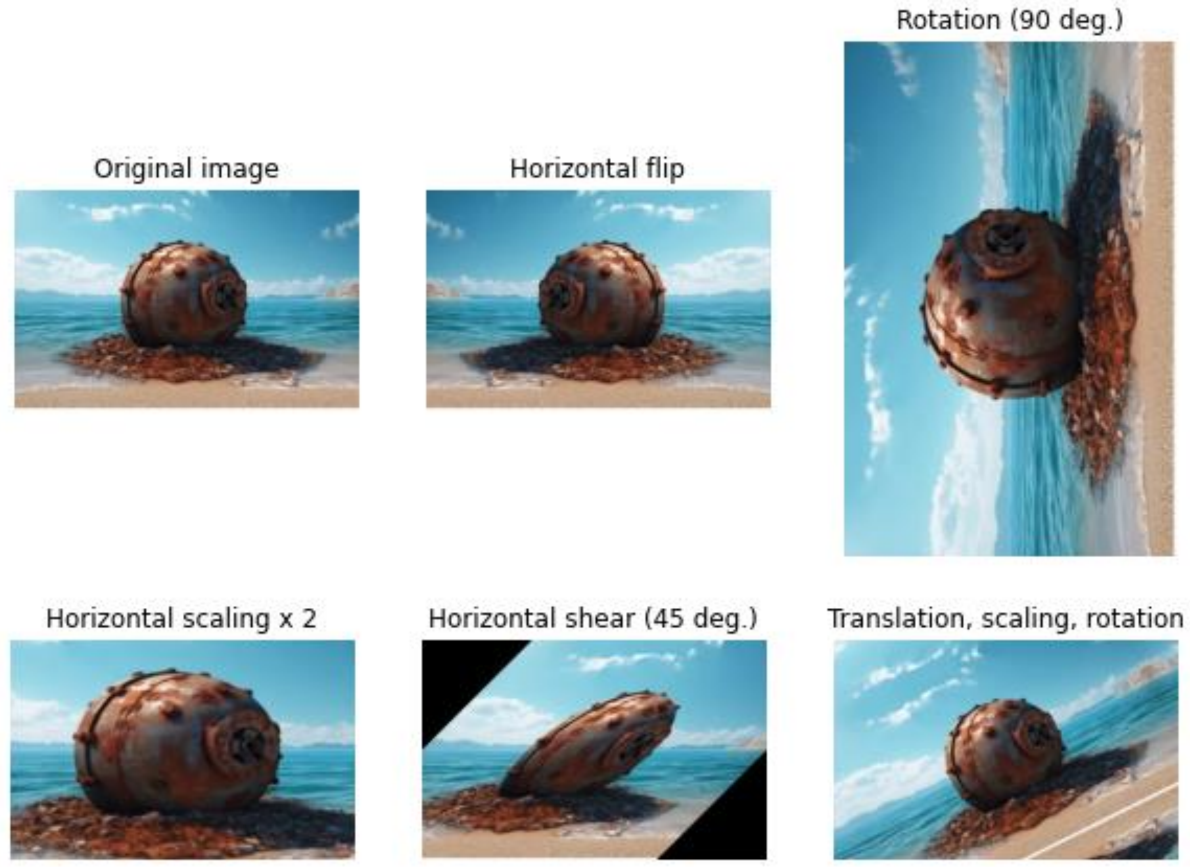

Figure 2. Affine transforms

Filtering transforms are of two main types: *high pass (sharpening) filters* and *low pass (smoothing) filters*. High pass filters, such as Contrast Limited Adaptive Histogram Equalization (CLAHE), sharpen edges and thus enhance contrast. On the other hand, low-pass filters can be used either for denoising or for simulating certain mechanical or optical effects. For example, they may simulate the way images appear blurred because of being taken through a glass, or because the camera went out of focus, or it moved while taking the picture. A selection of filtering transforms applied to an image of a sea mine is shown in Figure 3.

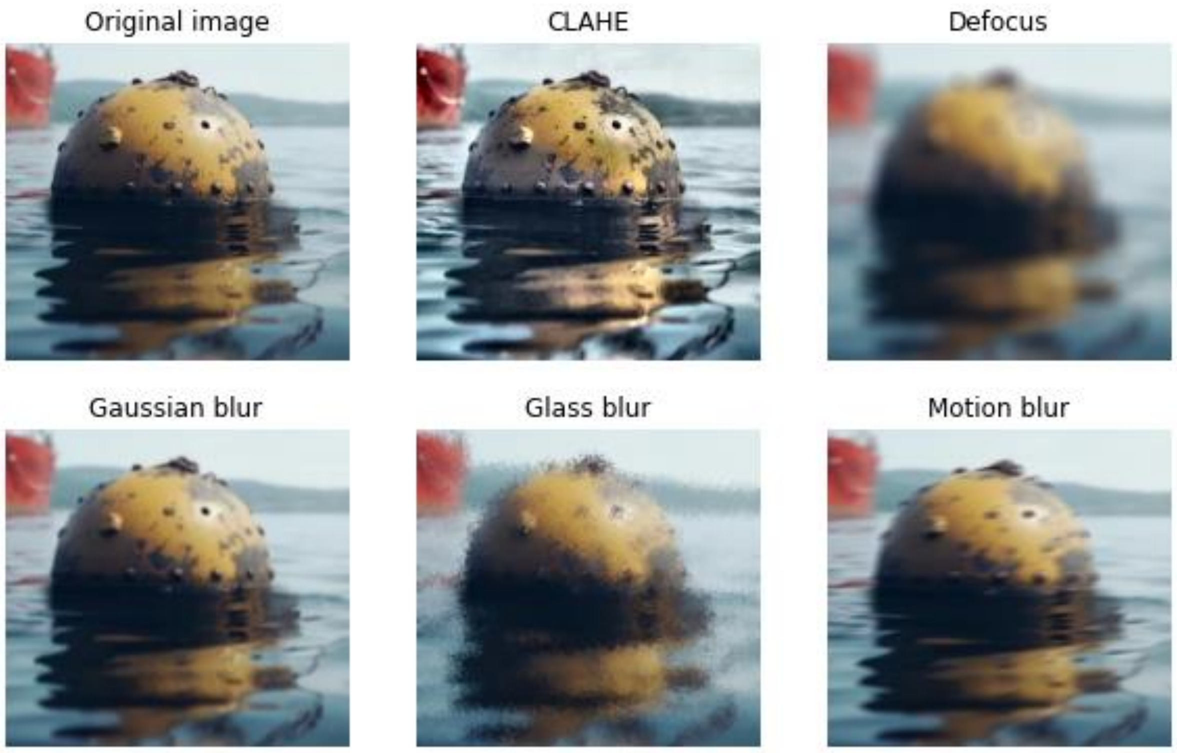

Figure 3. Filtering transforms

As for the "functional" transforms, they can simulate weather conditions (fog, with haze circles, rain, sun flare, shadows, snow, etc.), or photographical effects (haze circles under foggy conditions, camera sensor noise, solarization because of overexposure to light, etc.). Figure 4 illustrates several such effects.

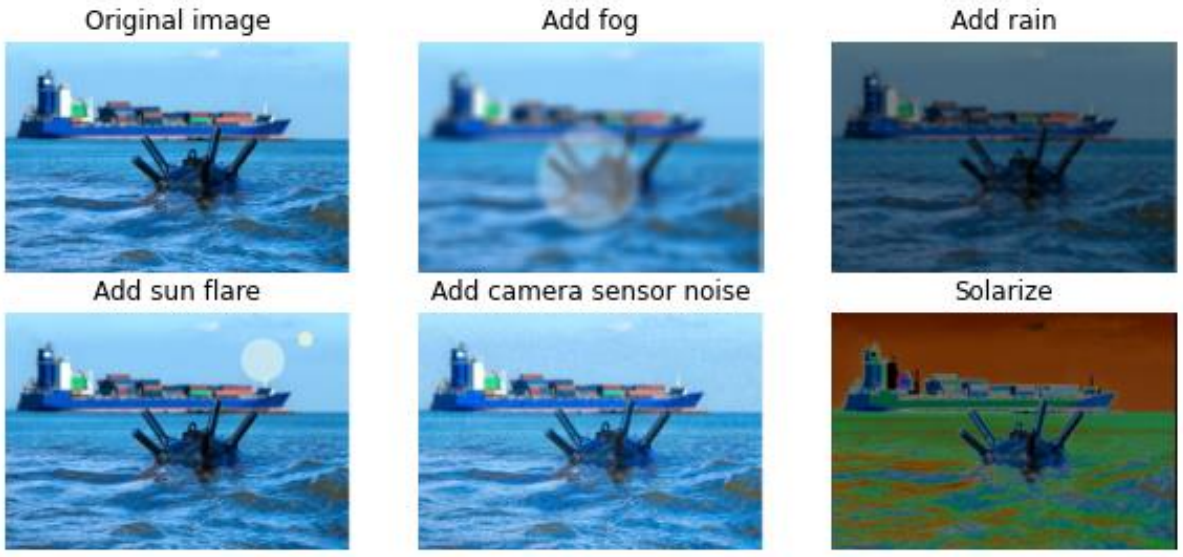

Figure 4. Functional transforms

It should be noted that, depending on the application and the technology used for data collection and processing, some augmentations may be useful, while others may be detrimental to the training process. The key to selecting suitable augmentations is similarity with real data that may constitute input to the model.

4. Object detection. The YOLO algorithm

Object detection is a computer vision task which consists of locating objects of given classes in images. A brief review of detection methods identifies two major approaches to object detection [2], [3]:

- Two stage approaches, which use one model for region proposal, and a second model for classifying the objects in the regions where they were detected. Examples are R-CNN, Fast R-CNN, Faster R-CNN, etc.
- One-stage approaches, which use a single model for both these tasks (region proposal and object classification). A popular example is YOLO (*You Only Look Once*). While the primary advantage of one-stage detectors over two-stage detectors is speed, YOLO has also been proven to outperform many competing approaches in terms of accuracy [6] [7].

For each object it detects in an input image, YOLO outputs the coordinates of the bounding box corresponding to the region of the detected object and the detection confidence scores for the classes of objects the model was trained to detect. An example is shown in Figure 5, where two objects were detected, with confidence scores of 82% and 87% for the class labels "ship" and "mine", respectively.

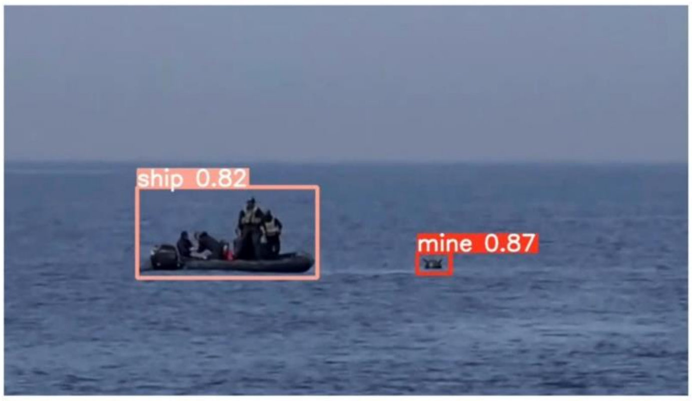

Figure 5. Object detection with YOLO

A typical example of a complete object detection pipeline with YOLO is presented in Figure 6. The first step is dataset creation, with augmentation if necessary. The next step is annotation (manual or other annotation/labelling tools). The next step is splitting the annotated dataset into the training, validation and testing parts. Next, the dataset should be exported to a format interpretable by YOLO. (for each image, there should be a text file containing the detected object classes and the normalized coordinates of the corresponding bounding boxes that surround the detected objects). Next, a configuration file in the YAML format should be edited. If a pretrained model is available, training may start from the parameters of that model, fine-tuning them. If not, it will start from random parameters. The final step is testing the trained model on input data from various sources (saved image or video files or collections of such files, camera input, etc.).

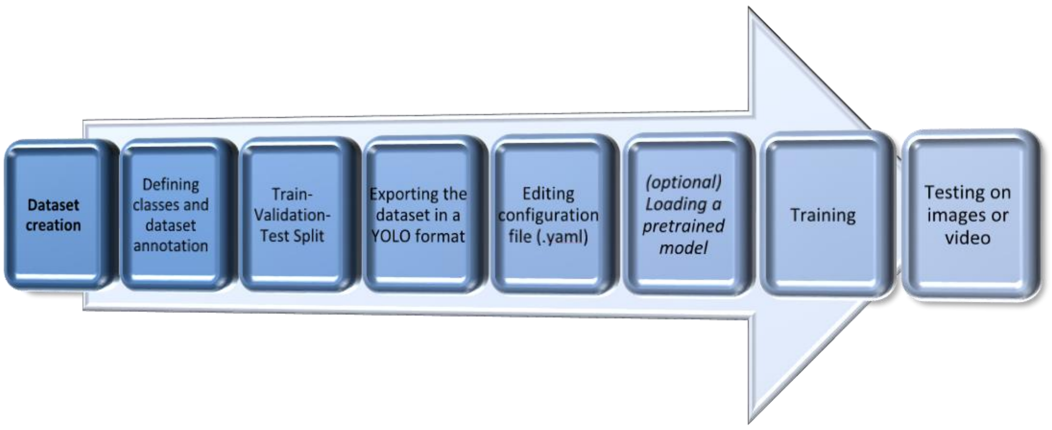
Dataset creation
Defining classes and dataset annotation
Train-Validation-Test Split
Exporting the dataset in a YOLO format
Editing configuration file (.yaml)
(optional) Loading a pretrained model
Training
Testing on images or video

Figure 6. Example computer vision pipeline with YOLO

5. Case study: AI Applied in the ASMINES Project

5.1 ASMINES project main stages

The ASMINES project stand in the development and integration of autonomous means such as (UAV and USV) in the concept of integrated action of specialized forces in the search and identification of sea mines.

The project that extends over a period of 3 years and has a series of main objectives that we will expose in the following.

The main objectives for the year 2023 were to develop the ASMINES concept and structure (SWOT analysis: fixed wing UAV vs multirotor UAV), purchase 3 UAV (2+1), the USV acquisition and conversion but not last the development of the seamines prediction trajectory and detection algorithm.

The first year project was ended with real-world testing of the USV component of the ASMINES system.

For the next two years (2024-2025) the research team has planned activities such as retesting in the real environment of the entire ASMINES system (UAV +USV) followed by testing the integrated UAV+USV action but this time launched from a command ship.

After that the final stage will consist of the approval of the prototype and the transfer of the system to the Romanian Naval Forces and the dissemination of project results

5.2 AI in the ASMINES project context

AI integration offers several significant benefits. It enhances accuracy by improving the precision of mine detection and minimizing false positives and negatives. It also boosts efficiency by speeding up the detection process, allowing for broader area coverage within shorter time frames. Additionally, AI enhances safety by reducing the need for human divers in initial mine detection, thereby increasing operational safety.

Looking ahead, the future directions involve ongoing development to improve AI algorithms for better accuracy and adaptability to various environments. There will also be efforts to integrate more advanced sensors and data fusion techniques to further enhance detection capabilities.

In terms of detection, the YOLO algorithm can identify sea mines based on a trained model. Upon successful detection (Fig. 7), the application records the coordinates of the aerial drone's position at the time of finding the sea mine in the database.

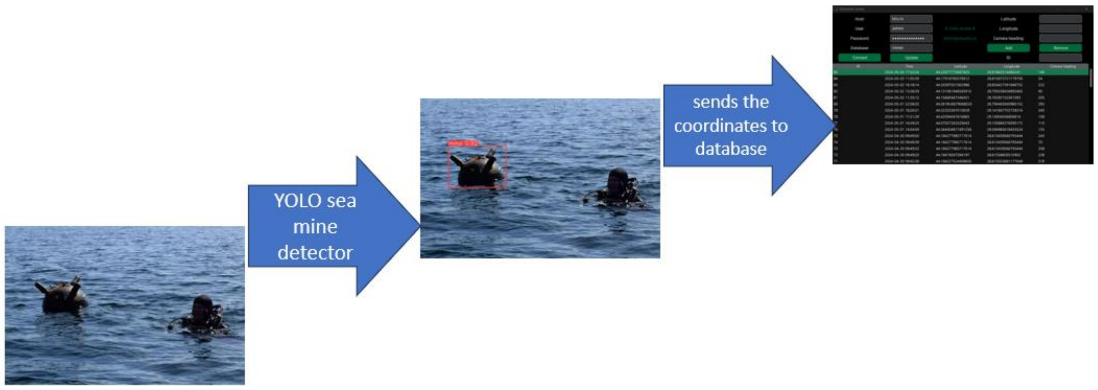

Fig. 7 - the logical flow of the algorithm in the ASMINES project context

6. Conlusions

The article presents, in an original conception, the possibility of using AI technologies in the process of identifying and classifying drifting sea mines. The fundamental theoretical aspects that were the basis of the design of the software application dedicated to this project were validated by the tests carried out in the operational environment, based on the information acquired from the sensors placed on the USV surface drone. We note the precision and speed with which the sea mine is identified, with the observation that detection performance is strictly dependent on the training of the identification algorithm and the number of images (real or augmented) of the intended target.
The practical results obtained will be capitalized by the realization of the ASMINES experimental system, which will be completed in 2025.

7. References

[1] Y. Wang, Z. Liu and S. Lian, "Semi-supervised Object Detection: A Survey on Recent Research and Progress.," arXiv preprint arXiv:2306.14106, 2023.

[2] Z. Q. Zhao, P. Zheng, S. T. Xu and X. Wu, "Object detection with deep learning: A review," *IEEE Transactions on Neural Networks and Learning Systems,* vol. 30, no. 11, pp. 3212-3232, 2019.

[3] Z. Zou, K. Chen, Z. Shi, Y. Guo and J. Ye, "Object detection in 20 years: A survey," *Proceedings of the IEEE,* vol. 111, no. 3, pp. 257-276, 2023.

[4] S. Rajvanshi, G. Kaur, A. Dhatwalia, S. A. Arunima and A. Bhasin, "Research on Problems and Solutions of Overfitting in Machine Learning," in *International Conference on Artificial-Business Analytics, Quantum and Machine Learning*, Singapore, 2023.

[5] A. Buslaev, V. I. Iglovikov, E. Khvedchenya, A. Parinov, M. Druzhinin and A. A. Kalinin, "Albumentations: fast and flexible image augmentations," *Information,* vol. 11, no. 2, pp. 125-128, 2020.

[6] J. Terven and D. Cordova-Esparza, "A comprehensive review of YOLO: From YOLOv1 to YOLOv8 and beyond," arXiv preprint arXiv:2304.00501, 2023.

[7] C. Y. Wang, I. H. Yeh and H. Y. M. Liao, "YOLOv9: Learning What You Want to Learn Using Programmable Gradient Information," arXiv preprint arXiv:2402.13616, 2024.

[8] M. DA, “The mine warfare cycle: History, Indications, and Future,” September 2015. [Online]. Available: http://www.globalsecurity.org/military/library/report/1997/Morris.

[9] J. Rios, “Naval mines in the 21st century can NATO navies meet the challenge?,” 2005.

[10] “US Navy Maritime Mammal Program,” [Online]. Available: http://www.public.navy.mil/spawar/Pacific/71500/Pages/default.aspx. [Accessed September 2015].

[11] N. D. Marian, “Studii privind evaluarea resurselor de energie regenerabilă în zona litoralului românesc al Mării Negre,” [Online]. Available: http://arthra.ugal.ro/handle/123456789/7019.

[12] P. Bosscher and S. Lightsey, Unmanned surface vehicles for sea mine detection.

[13] S. A. Bendersky, YOLO-based detection of sea mines in underwater video, 2020.

[14] H. S. Kim, Real-time detection and tracking of underwater mines using unmanned aerial vehicles.

[15] D. Munteanu, D. Moina, C. G. Zamfir, Ș. M. Petrea, D. S. Cristea and N. Munteanu, “Sea Mine Detection Framework Using YOLO, SSD and EfficientDet Deep Learning Models,” [Online]. Available: https://www.mdpi.com/1424-8220/22/23/9536.