https://orcid.org/0009-0000-2986-509X
https://orcid.org/0000-0001-8163-1119
https://orcid.org/0000-0003-0348-5456

UDC 004.8

M.D. KAMAK[*], V.V. KAZYMYR[**], D.O. KAMAK[***]

METHOD OF DETECTION THE QUADCOPTERS AND OCTOCOPTERS BASED ON YOLOV8 MODEL

[*]Institute of Mathematical Machines and Systems Problems of the National Academy of Sciences of Ukraine, Kyiv, Ukraine
[**]Chernihiv Polytechnic National University, Chernihiv, Ukraine
[***]State Scientific Research Institute of Armament and Military Equipment Testing and Certification, Cherkasy, Ukraine

Анотація. *Актуальність теми статті визначається все більшою поширеністю безпілотних літальних апаратів (БПЛА) не тільки в цивільній, а й у військовій сфері, де вони створюють значні проблеми протиповітряної оборони. У роботі представлено практичну перевірку можливості та ефективності використання моделі YOLOv8 для виявлення квадрокоптерів та октокоптерів у відеопотоках, яка призначена для виявлення об'єктів у реальному часі та сегментації зображення. Використовується метод навчання, який адаптує модель YOLOv8 шляхом застосування підходу «transfer learning» для коригування попередньо навчених вагових коефіцієнтів YOLOv8 до даних, що стосуються дронів. Також досліджується здатність моделі ефективно працювати в різних умовах зовнішнього середовища. Проведені експерименти демонструють ефективність розробленого методу для точної ідентифікації БПЛА в різних ситуаціях, що робить його важливим для вдосконалення засобів повітряного спостереження та механізмів безпеки. Дослідження також аналізує адаптивність моделі до мінливих умов спостереження, забезпечуючи її надійність при обробці зображень і відео. Результати вказують на високий потенціал моделі YOLOv8 у підвищенні можливостей систем протиповітряної оборони та зміцненні заходів безпеки проти загроз БПЛА. Крім того, у статті обговорюється обчислювальна ефективність моделі, підкреслюється її здатність обробляти відеопотоки в реальному часі з мінімальними обчислювальними ресурсами. Оцінено показники precision та recall моделі, що демонструє її здатність точно виявляти БПЛА з мінімізацією помилкових спрацьовувань. Загалом результати підкреслюють важливість використання передових методів машинного навчання для ефективного виявлення БПЛА.*
Ключові слова: *безпілотні літальні апарати, YOLOv8, квадрокоптери, октокоптери, протиповітряна оборона, обробка відеопотоку, виявлення об'єктів, спостереження в реальному часі, transfer learning, обчислювальна ефективність, точність і запам'ятовування, виявлення дронів.*

Abstract. *The relevance of the topic of the article is determined by the growing prevalence of unmanned aerial vehicles (UAVs) not only in the civilian but also in the military field, where they create significant problems for air defense. The paper presents a practical test of the possibility and effectiveness of using the YOLOv8 model for detecting quadcopters and octocopters in video streams, which is designed for real-time object detection and image segmentation. A training method is used that adapts the YOLOv8 model by applying a «transfer learning» approach to adjust pre-trained YOLOv8 weights to drone-specific data. The ability of the model to work effectively in various environmental conditions is investigated, too. The conducted experiments demonstrate the efficiency of the developed method to accurately identify UAVs in various situations, which makes it important for improving aerial surveillance tools and security mechanisms. The study also explores the model's adaptability to changing observation conditions, ensuring its robustness in processing images and video. The results indicate the high potential of the YOLOv8 model in enhancing the capabilities of air defense systems and bolstering security measures against UAV threats. Additionally, the article discusses the computational efficiency of the model, highlighting its ability to process video streams in real time with minimal computational resources. The precision and recall metrics of the model are evaluated, demonstrating its ability to accurately detect UAVs while minimizing*

false positives. Overall, the findings underscore the importance of leveraging advanced machine-learning techniques for effective UAV detection.
Keywords: *unmanned aerial vehicles, YOLOv8, quadcopters, octocopters, air defense, video stream processing, object detection, real-time surveillance, transfer learning, computational efficiency, precision and recall, drone detection.*

DOI: 10.34121/1028-9763-2024-2-65-77

1. Introduction

In the modern era, the sky has become a bustling highway for an ever-increasing fleet of unmanned aerial vehicles (UAVs), popularly known as drones. Their ascent has been meteoric, finding roles in a broad array of activities from capturing breathtaking aerial photography to being the main actor in modern war conflicts (the Russo-Ukrainian war).

The rapid growth and development of drone technology have raised the issue of countering drones [1, 2]. For example, in our current realities, enemy military drones are constantly conducting aerial reconnaissance of our positions, making kamikaze raids on personnel and equipment, and no less importantly, adjusting artillery fire. All of these emphasize the need for reliable detection systems for small and medium-sized aircraft [3–6]. The detection challenge is heightened by the diversity of different types of drones, such as quadcopters and octocopters, each with unique flight patterns and risk profiles.

Into this breach steps YOLOv8 [7], the latest iteration in the acclaimed «You Only Look Once» series, lauded for its lightning-fast object detection capabilities. This paper delves into the utilization of YOLOv8 as a sentinel in the sky, tasked with the identification and tracking of these UAVs in real-time video feeds. We chart the model's learning trajectory, elucidated by the training graphs, and scrutinize its field performance through a series of test set examples.

The aim of the article is to merge cutting-edge machine-learning techniques with the pressing need for aerial oversight. We explore the potential of YOLOv8 to not just see but to understand the skies.

2. Problem statement

The proliferation of drones presents a multifaceted challenge in modern aerial defense, particularly in conflict zones where they play pivotal roles in reconnaissance, offensive operations, and artillery coordination. Conventional detection systems often struggle to identify and track these agile machines, particularly smaller models like quadcopters and octocopters that can maneuver with a low radar cross-section. YOLOv8 offers a potential solution with its advanced object detection algorithms capable of processing video streams in real time [8]. The problem at hand is developing a model using YOLOv8 that can accurately detect diverse drone types in various operational scenarios, ensuring robust defense mechanisms against UAV threats.

3. Methodology

Our methodology harnesses the capabilities of YOLOv8 for drone detection in video streams. The YOLOv8 model, an advanced iteration in the «You Only Look Once» series, was selected for its computational efficiency and high precision in object detection tasks. Our approach utilized a transfer learning paradigm, initializing YOLOv8 with weights pretrained on a comprehensive dataset to capitalize on pre-established feature detection capabilities. Subsequent fine-tuning on a drone-specific dataset, inclusive of various UAV classes and contextual backgrounds, aimed to enhance the model's discriminatory power. The fine-tuning process involved meticulous hyperparameter optimization, seeking a parsimonious balance between detection latency and accuracy. Validation was conducted through the stratified k-fold cross-validation to assess generalizability across unseen data. This iterative training approach, underpinned by continuous performance

monitoring via loss metrics and mAP scores, ensured the refinement of the model toward optimal operational efficacy.

In this paper, we focus on detecting drones with an average size of 347.5×283×107.7 mm (L×W×H) and a speed not exceeding 15 m/s. The chosen input parameters of object detection are since the most common drone on the battlefield «DJI MAVIC 3» has such characteristics. We use the IMX219 camera module with a 3264×2464 sensor resolution. Thus, we use 100 m as the average possible detection distance. We tried to fill our dataset with images of different types of environments, from empty clear skies to urban landscapes, to test the efficiency of the model in different environmental scenarios and to analyze which of them is the most favorable and in which one the model has difficulties in the detection process.

4. YOLOv8 architecture and functionality

YOLOv8, an advanced iteration in the YOLO series, utilizes a deep convolutional neural network (CNN) for efficient object detection. The architecture comprises three main components: the backbone, the neck, and the head [9].

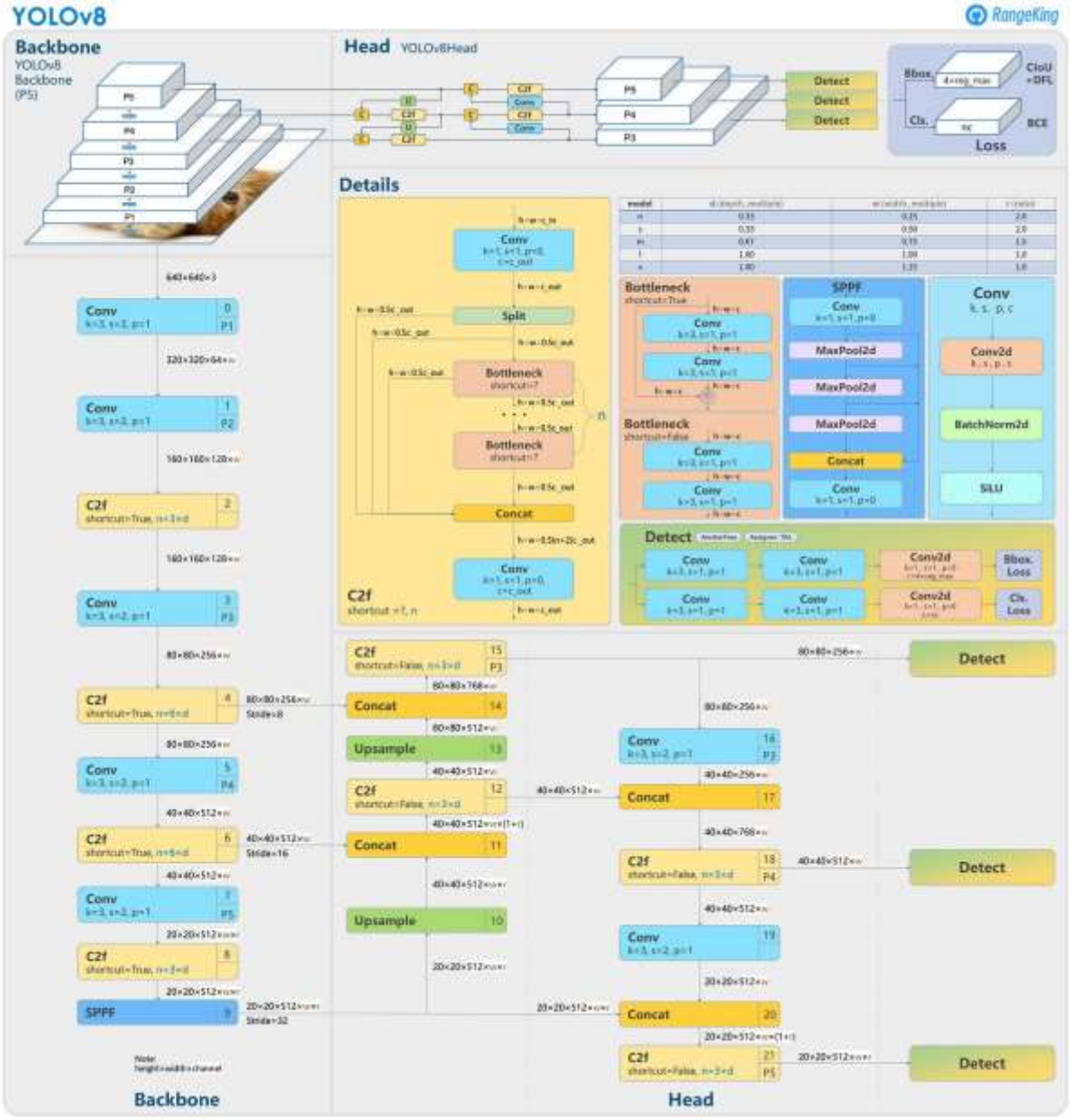

Figure 1 — Main components of the YOLOv8 architecture [9]

– The backbone is designed for feature extraction. It uses a series of convolutional layers to process the input image and extract a set of feature maps at different scales. This part of the network is critical for identifying various attributes of objects within the image.

– The neck connects the backbone to the head. It processes the feature maps from the backbone, refining and recombining them. This step is crucial for preparing the features for precise object detection.

– The head of the network is responsible for making predictions. It uses the processed feature maps to predict bounding boxes, object classes, and abjectness scores. The head employs a set of anchor boxes and the predefined shapes that help the model detect objects of various sizes and shapes efficiently.

You can find the described parts in the picture below (see Fig. 1).

This architecture consists of 53 convolutional layers and employs cross-stage partial connections to improve information flow between different layers [9].

4.1. Method of training the YOLOv8 model to detect UAV

The development of a robust UAV detection system via YOLOv8 entailed a systematic training methodology, initiated by curating a dataset using the Roboflow platform [11], which streamlined the collection and pre-processing of diverse images of drones (see Fig. 2). The platform provides a convenient UI to automate common steps (e.g., collecting training data, assigning classes, and annotating objects) for training object detection models. The figures below demonstrate how you can use the platform UI to perform the steps mentioned above.

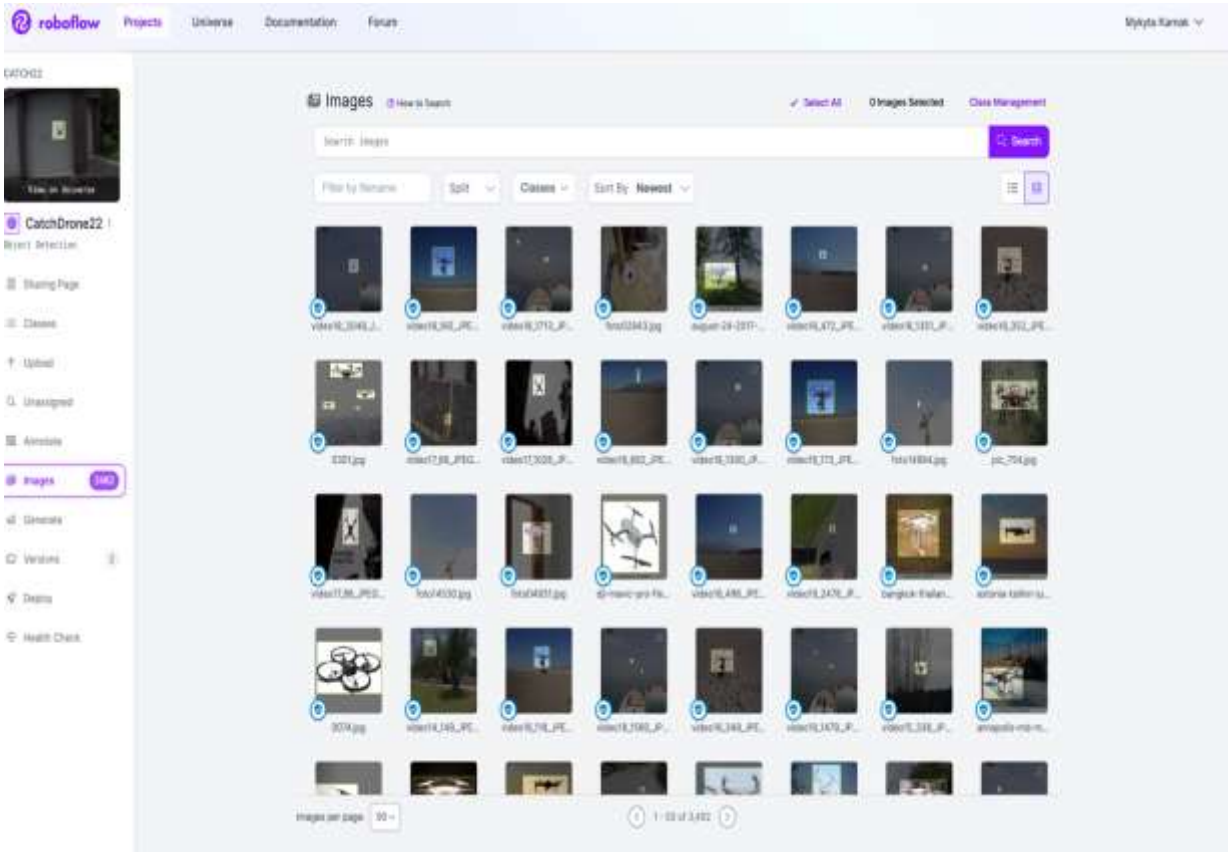

Figure 2 — Drone images within drones dataset located in Roboflow “images” UI

In object detection tasks, it can be advantageous to start with a singular classification category when there is insufficient data to reliably differentiate between subcategories or when the

distinctions are not critical to the project's goals. It simplifies the model and reduces the complexity of the task, which can be particularly useful at the early stages of model training or when the focus is on detecting the presence of any drone as in our case (see Fig. 3).

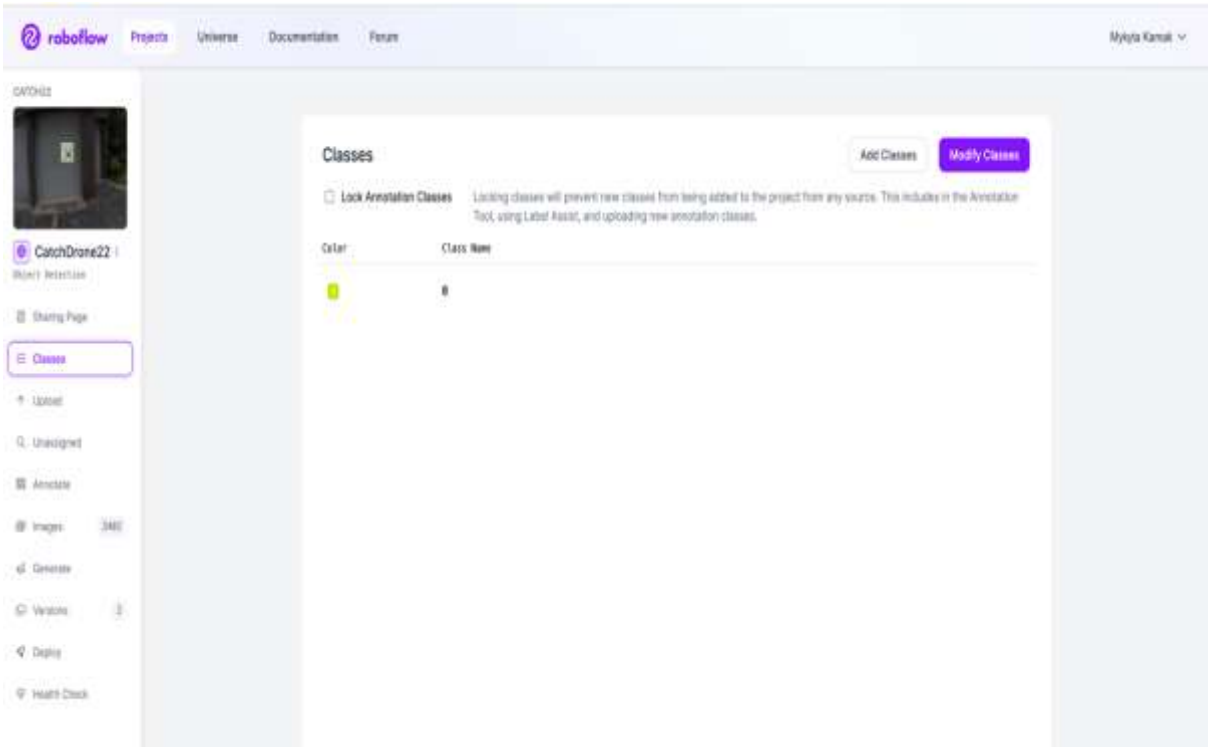

Figure 3 — Detection class located in Roboflow "Classes" UI

The last step in preparing the training dataset for our model is annotation of objects we want to detect. Therefore, each previously uploaded drone image was carefully annotated to ensure accurate training of the model (see Fig. 4).

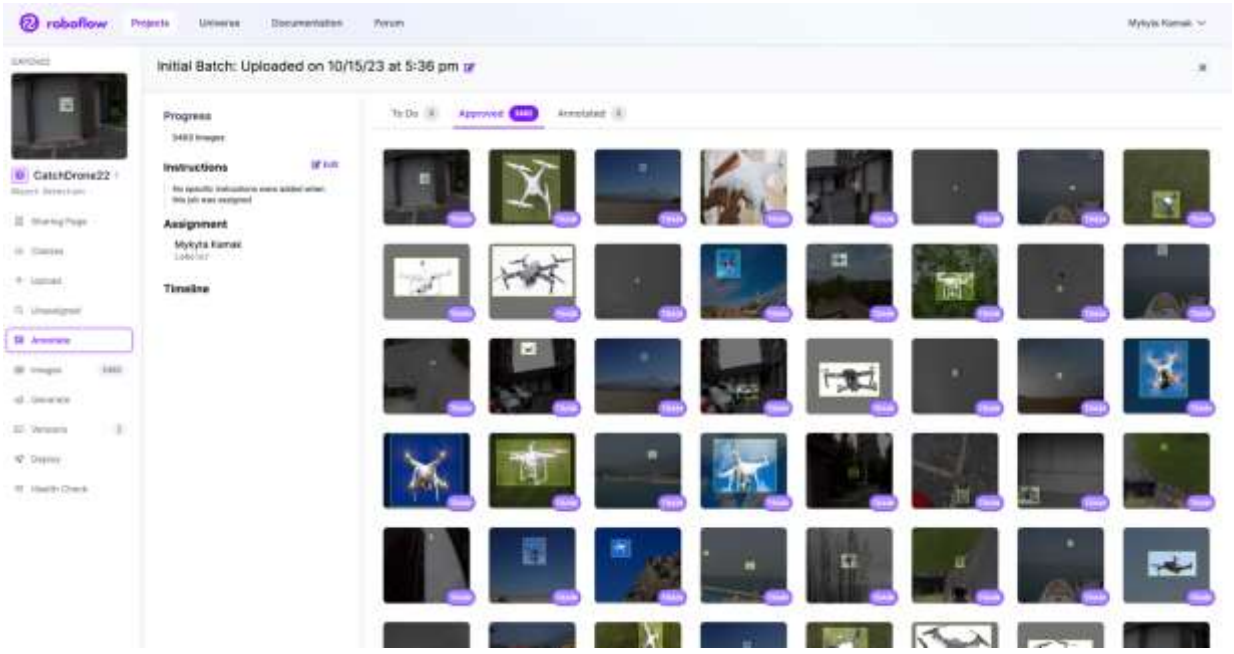

Figure 4 — Annotation functional of the Roboflow platform

The training process, scripted in a Python notebook [12], utilized the Roboflow library to retrieve the dataset and implemented a series of commands to train the YOLOv8 model. The key parameters such as the number of epochs and imgsz (image size) were configured to optimize model performance for real-world applications (see Fig. 5).

The term «epoch» in machine learning refers to one complete cycle through the full dataset during training. In each epoch, the model learns by adjusting its weights to minimize the error in its predictions. The number of epochs is a hyperparameter that determines how many times the learning algorithm will work through the entire training dataset. Choosing the number of epochs typically involves a trade-off: too few may result in an underfit model, while too many can lead to overfitting, where the model learns the training data too well, including noise and outliers, and performs poorly on new, unseen data. We choose 100 epochs in our settings because it is recommended by the YOLOv8 developers' value.

The image size (imgsz) is also an important hyperparameter in object detection models. It determines the resolution at which the input images are processed. The size of 640x640 pixels is chosen because it is a common standard that balances the need for detail (to detect and classify objects accurately) with computational efficiency (to train and run the model quickly). It is large enough to capture relevant features of the objects but not so large that it would require excessive computational power or memory, which could slow down the training process or make the model less deployable in environments with limited resources that is very important for us, because as general aerial surveillance tools need to be portable, which imposes a resource constraint on them.

```
[ ] !yolo task=detect \
    mode=train \
    model=yolov8s.pt \
    data={dataset.location}/data.yaml \
    epochs=100 \
    imgsz=640
```

Figure 5 — Key parameters of the training process

The complete pipeline of model training consists of the following steps:
- Data collection: acquire diverse drone images.
- Data annotation: manually annotate images with bounding boxes.
- Data preprocessing: resize, normalize, and augment data using Roboflow.
- Model configuration: set the YOLOv8 parameters and hyperparameters.
- Model training: train YOLOv8 with the prepared dataset.
- Model evaluation: validate using metrics like mAP, precision, and recall.
- Model optimization: tune parameters based on validation results.
- Deployment: deploy the trained model for real-time detection.

5. Results of the YOLOv8 model training

The results of the model training process for the detection of UAVs, specifically quadcopters and octocopters, include both quantitative and qualitative ones and are divided into two samples: training and validation. Our full dataset of 8 220 images (4 349 images of quadcopters and 3 871 images of octocopters) was divided into samples in the following ratio: 60 % for training, 20 % for validation, and 20 % for testing, according to the best practices in AI training for small sets (up to a few thousand examples).

The quantitative results comprise a series of graphs that illustrate the model's performance over the training period (see Fig. 6). Prefix «train/» in graph titles represents the training part and «val/» — the validation one. For all the charts, the X-axis represents the number of

epochs. Y-axis represents different values depending on the graph type (see Formula 1, 2, 3, 4, 5, and 6).

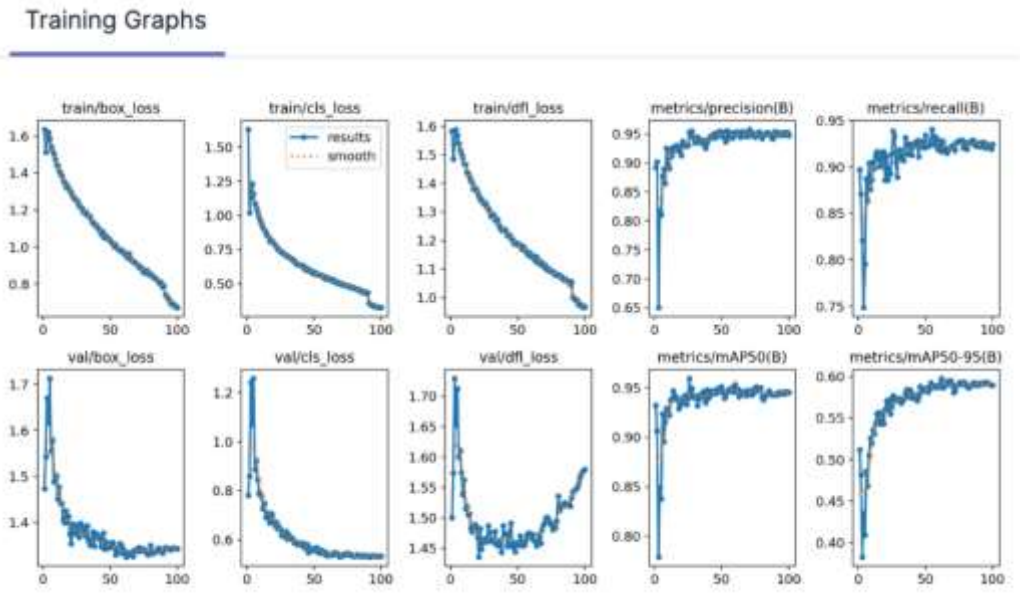

Figure 6 — Graphs of the model performance over the training period

Loss Metrics. The graphs for *train/box_loss*, *train/cls_loss*, and *train/df1_loss* alongside their validation counterparts (*val/box_loss, val/cls_loss,* and *val/df1_loss*) exhibit a typical downward trend, indicative of the model's improving ability to correctly identify and classify objects within the training dataset. As the number of epochs increases, the smooth lines illustrate a consistent reduction in loss, reflecting the model's increasing accuracy.

The *box_loss* graphs calculate the average squared difference between the predicted bounding box coordinates and the true bounding box coordinates across all N predictions. Each prediction i involves comparing the vector of predicted coordinates y_{pred_i} (such as the center, width, and height of a box) against the true coordin ates y_{true_i}. It is important because it directly measures how accurately the model predicts the location and size of the bounding boxes in object detection tasks. Minimizing this loss is crucial for improving the precision of object localization in images.

$$\text{Box Loss} = \frac{1}{N}\sum_{i=1}^{N}\left(y_{\text{true}_i} - y_{\text{pred}_i}\right)^2. \quad (1)$$

The *cls_loss* graphs are calculated based on Cross-Entropy Loss function. It calculates the average negative log probability of the correct class C across all N examples. Here $y_{\text{true}_{i,c}}$ is a binary indicator (0 or 1) if class label C is the correct classification for observation i, and $y_{\text{pred}_{i,c}}$ is the predicted probability that observation i belongs to class C. The cross-entropy loss is crucial for the classification tasks as it penalizes incorrect class predictions. It is especially sensitive to confident wrong predictions, which is beneficial in training classifiers to output probabilities close to 0 for wrong classes and close to 1 for the correct class.

$$\text{Classification Loss} = -\frac{1}{N}\sum_{i=1}^{N}\sum_{c=1}^{C} y_{\text{true}_{i,c}} \log\left(y_{\text{pred}_{i,c}}\right). \quad (2)$$

Performance Metrics. In precision and recall graphs, *metrics/precision(B)* and *metrics/recall(B)* remain high throughout the training epochs, maintaining values close to 1 (0.94

based on the graphs). This suggests that the model has a high probability of correctly identifying UAVs when they are present and there is a low rate of false negatives. The Mean Average Precision (mAP) at different Intersections over Union (IoU) thresholds (*metrics/mAP50(B)* and *metrics/mAP50-95(B)*) are robust, which is critical for applications where the correct detection of UAVs is paramount.

Precision is a metric that calculates the ratio of correctly predicted positive observations to the total predicted positives (the sum of true positives (*TP*) and false positives (*FP*)) [13]. It is important because it shows how reliable the model's predictions are; the higher the precision, the more you can trust the model's positive predictions. It is particularly important in domains where the cost of false positives is high.

$$\text{Precision} = \frac{TP}{TP+FP}. \tag{3}$$

In the recall metric, *FN* stands for False Negatives — the count of positive cases that were incorrectly predicted as negative. This metric measures the ability of a model to find all the relevant cases (all positive samples).

$$\text{Recall} = \frac{TP}{TP+FN}. \tag{4}$$

The mAP at a single IoU threshold (like 0.50) is the mean of the Average Precision (AP) scores for all classes *Q*. AP for a single class is computed by integrating the area under the precision-recall curve for that class. mAP is an essential metric in object detection because it accounts for both the precision and recall of the predictions, providing a balanced view of the model's overall performance. For our model, it is close to 0.95.

$$\text{mAP} = \frac{1}{Q}\sum_{q=1}^{Q} AP_q. \tag{5}$$

This variant of mAP is averaged over a range of IoU thresholds, typically from 0.50 to 0.95 in increments (like 0.05). It calculates the average mAP across these thresholds, representing a more robust measure of the model's performance at different levels of bounding box overlap. This comprehensive metric is important as it ensures the model is consistently good across various degrees of detection strictness, which is valuable for practical applications where different conditions/environments may require different precision levels for the detected objects. Based on our testing, we obtained a value for this metric that is close to 0.59.

$$\text{mAP}_{\text{IoU=0.50:0.95}} = \frac{1}{T}\sum_{t=1}^{T} mAP_{\text{IoU}_t}. \tag{6}$$

5.1 Analyzing effectiveness in different environments

Qualitative results provide a visual affirmation of the model's real-world efficacy. The example outputs from the test set feature the model's predictions superimposed on images of various environments.

The ability of the model to detect drones in different environmental conditions is extremely important for its practical application in realistic systems, especially in the military sector, as the accuracy of the model can affect the outcome of a decision or the lives of personnel and the integrity of property. Therefore, we have conducted a number of tests. Below there is a photo report of some tests to illustrate the process, as well as a table describing the other tests that were not included in the photo report for reasons of keeping the material brief.

Fig. 7 demonstrates the interface of test system [14] for the trained model. The red arrows indicate that we have the ability to use images or video stream as input parameters of the model to test in in different environments. You can also access to file with weights for the model that we use in this paper at this link [15].

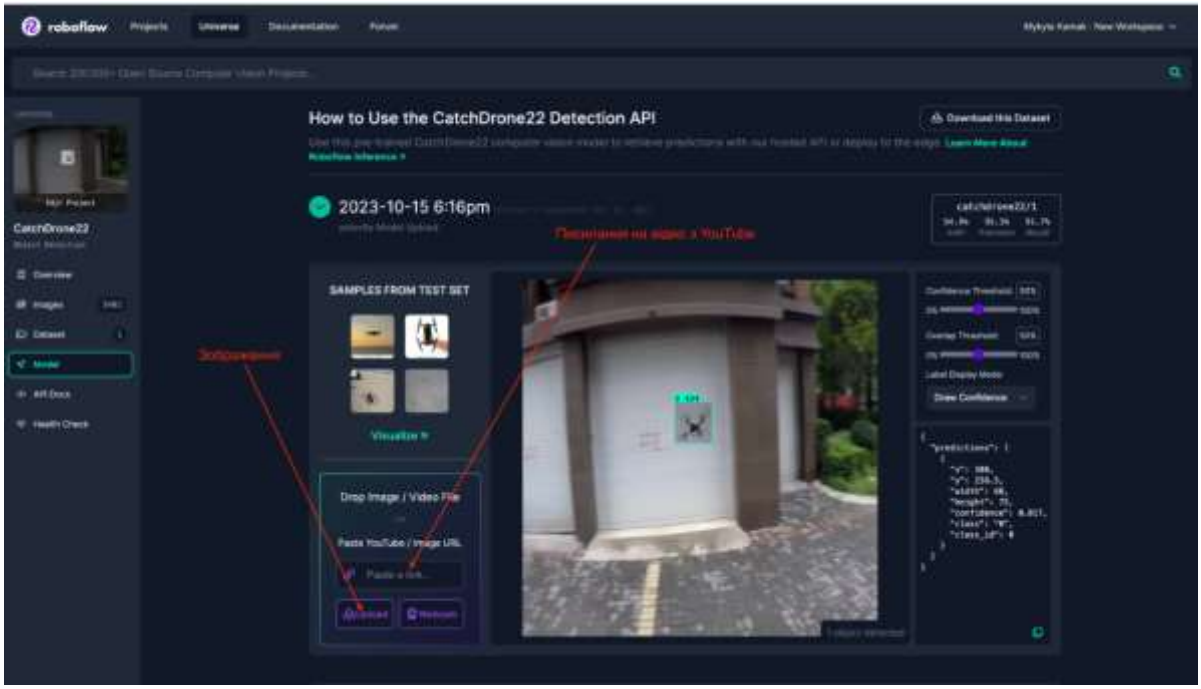

Figure 7 — Test system interface

Figures 8, 9, 10 and 11 are the photo report. Real values of such parameters as confidence, class of detection, and boxes coordinate you can find in each figure in the low right corner, in the red box.

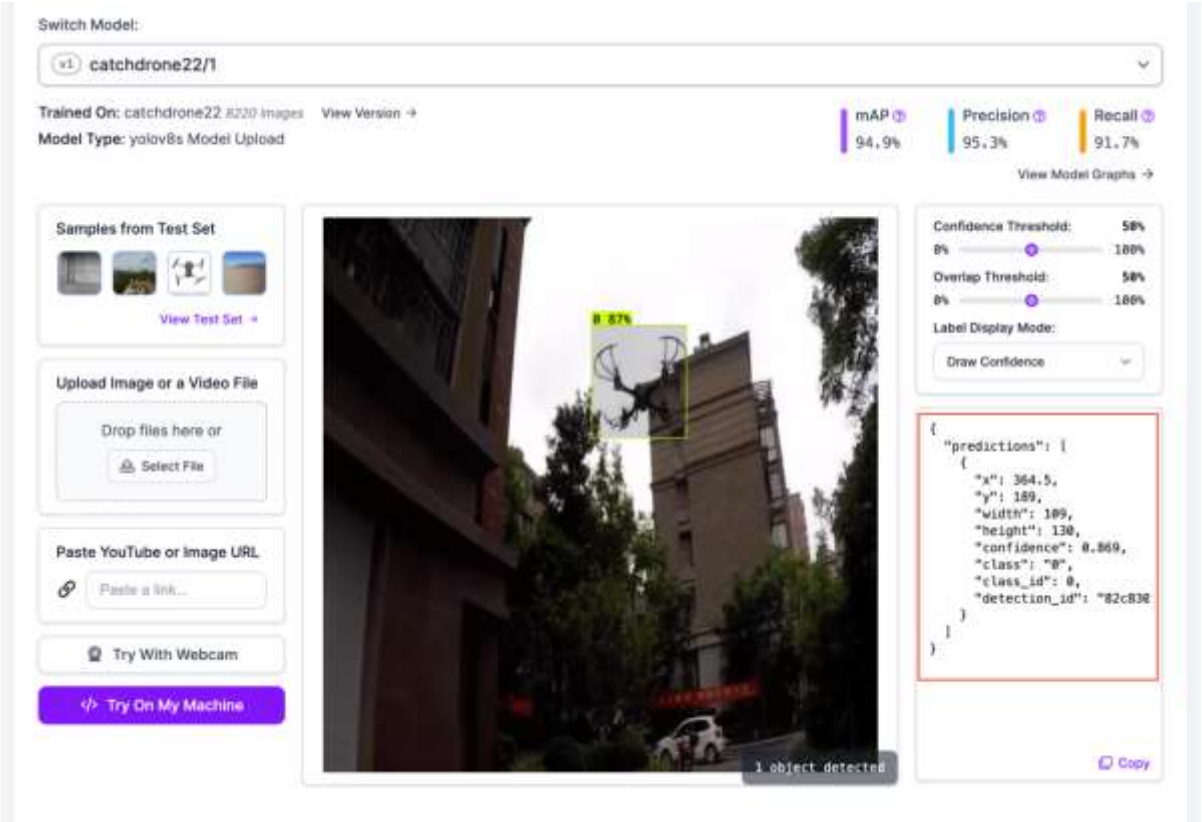

Figure 8 — Result of drone detection in an urban environment

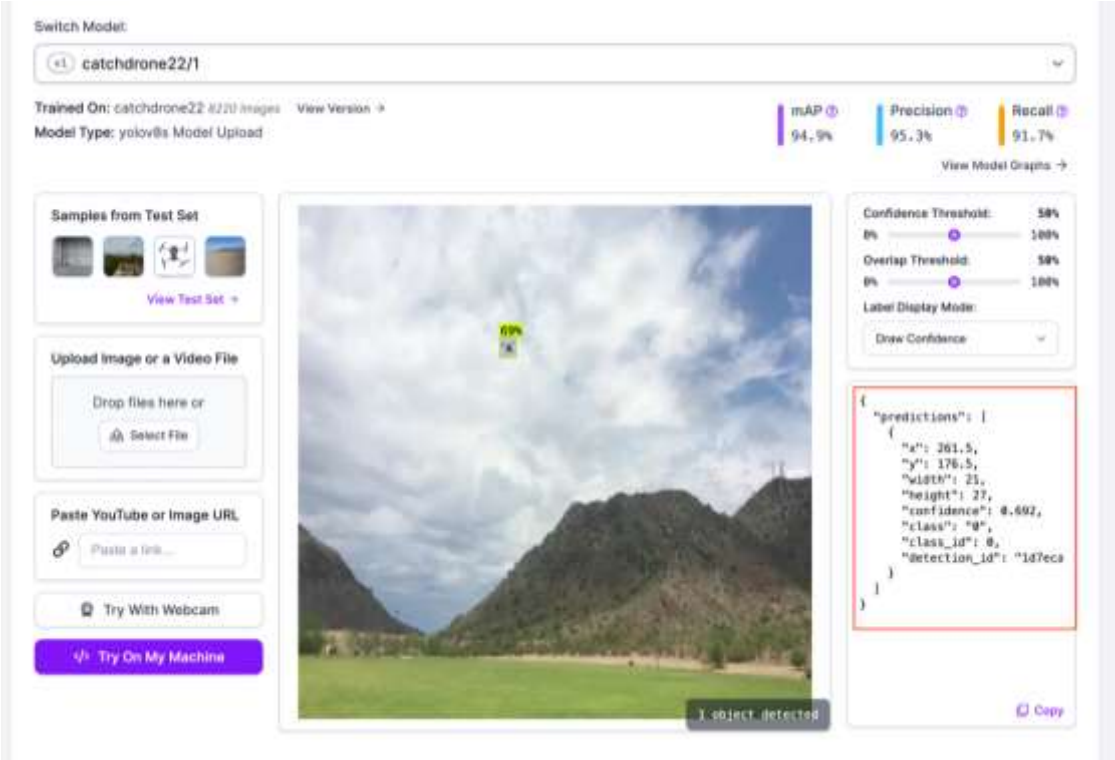

Figure 9 — Result of drone detection in a mountain environment

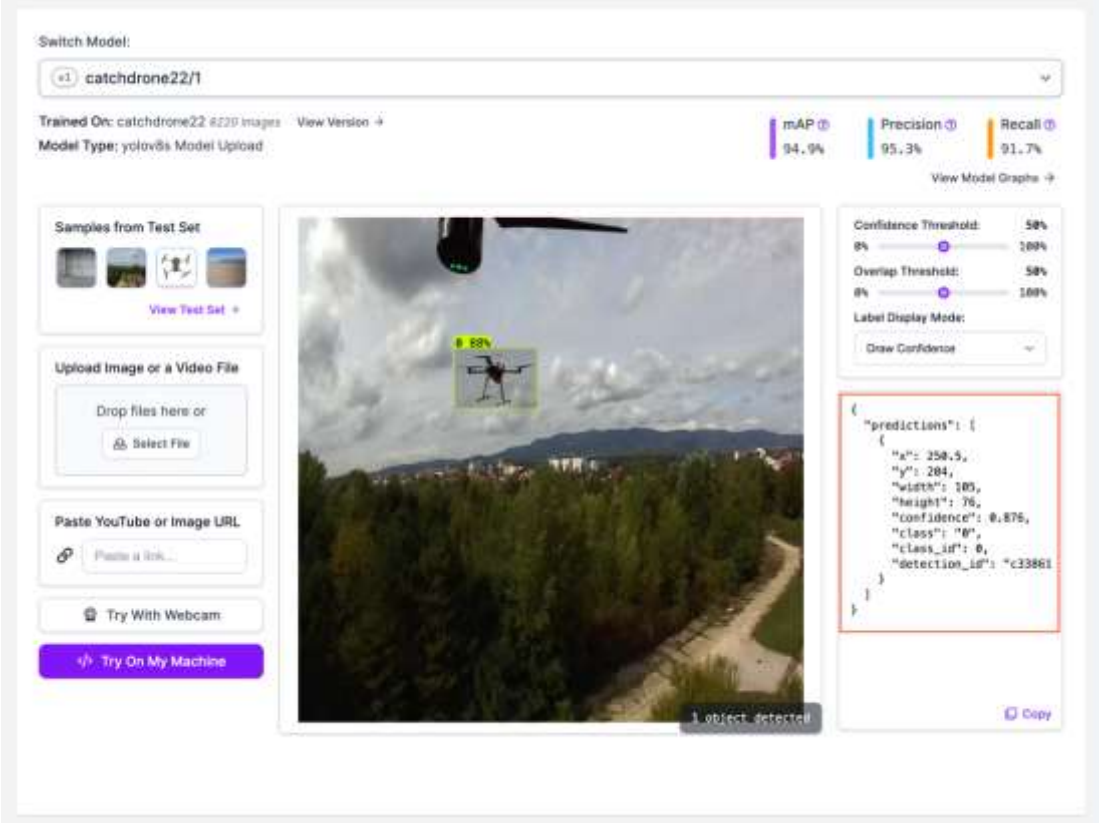

Figure 10 — Result of drone detection in a mix environment from another drone camera

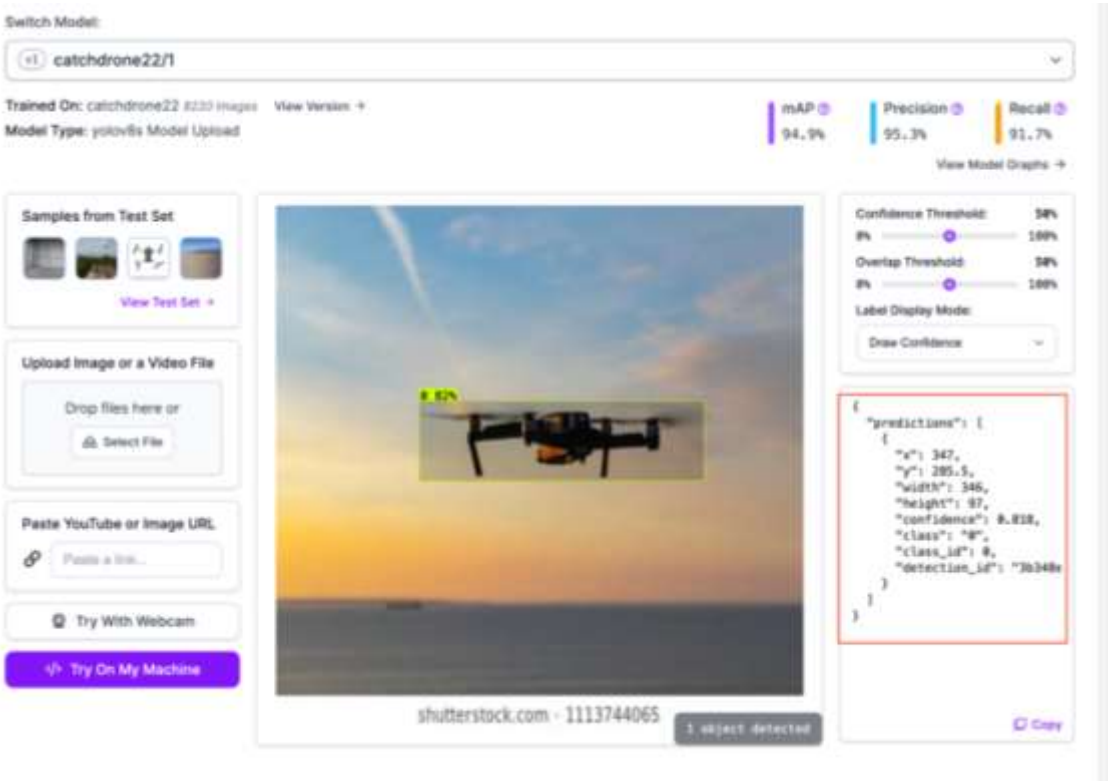

Figure 11 — Result of drone detection in a marine environment

Table 1 below describes the results of our tests for different environments. The «Environment» column contains the description of the environment itself. «Count of test» contains epy count of different images with the same environmental condition that was used in the test. «Avg confidence in %» represents the average result in per cents for confidence value that was obtained from the run of a set of tests based on the «Count of test» column.

Table 1 — Result of test in different environments

Environment	Count of tests	Avg confidence in %
Urban	30	81
Mountains	20	68
Mixed	50	79
Sky	200	89
Low light sky	100	53
Low light urban	20	65
Sea	20	82
Earth (the view from top to down)	40	73

Analyzing the results of testing, we can take notice of the following:

– The best result of detection we got in case when a drone was in high contrast background such as sky, sea, or earth. It is expected because in this case the model can find more features of an object and distinguish it from the background.

– The detection of drones on the earth background gives worse results than in the sky because in most of our test cases the drone has a black or dark color and it increases the chances that the drone's features would blend in with the color of the ground.

– The detection in low-light conditions was a real challenge for the model. We should think about how we can increase confidence in this environment in the next version of the model.

6. Conclusions

The application of YOLOv8 for UAV detection in video streams has demonstrated promising results. During the sequence of experiments, we have found out the balance between hyperparameters (such as epochs and imgsz) to get the best result on real-time data that was confirmed by the training graphs which have shown a consistent decrease in loss metrics, suggesting that the model has learned to detect UAVs effectively. Regardless of the type of drone design (quadcopter or octocopter), the precision and recall metrics are consistently high close to 0.95 and 0.92 corresponding, indicating a reliable model. Qualitative assessments through the test examples further validate the model's practical utility. The project underscores the potential of YOLOv8 in enhancing aerial defense mechanisms by accurately identifying UAV threats in diverse environments. Also, we found out that environments with high contrast backgrounds such as sky (~89% of detection confidence) or sea (~82% of detection confidence) have higher confidence values because it allows the model to detect object features easier than in environments with low light or uneven background. Further developing our image data set to include more types of environmental conditions and UAV classes should increase the precision of detection in complex scenarios and increase the value of $mAP_{IoU=0.50:0.95}$ metric especially in low-light condition.

REFERENCES

1. Taha B., Shoufan A. Machine Learning-Based Drone Detection and Classification: State-of-the-Art in Research. *IEEE Access.* 2019. N **7**. P. 138669–138682. DOI: https://doi.org/10.1109/ACCESS.2019.2942944.
2. Samaras S., Diamantidou E., Ataloglou D., Sakellariou N., Vafeiadis A., Magoulianitis V., Lalas A., Dimou A., Zarpalas D., Votis K. et al. Deep Learning on Multi Sensor Data for Counter UAV Applications-A Systematic Review. *Sensors.* 2019. N 19. P. 4837. DOI: https://doi.org/10.3390/s19224837.
3. Mendis G.J., Randeny T., Wei J., Madanayake A. Deep learning based doppler radar for micro UAS detection and classification. *Proc. of the MILCOM 2016. IEEE Military Communications Conference* (Baltimore, MD, USA, November 1–3 2016). Baltimore, MD, USA, 2016. P. 924–929.
4. Ganti S.R., Kim Y. Implementation of detection and tracking mechanism for small UAS. *Proc. of the International Conference on Unmanned Aircraft Systems (ICUAS)* (Arlington, VA, USA, June 7–10 2016). Arlington, VA, USA, 2016. P. 1254–1260.
5. Hommes A., Shoykhetbrod A., Noetel D., Stanko S., Laurenzis M., Hengy S., Christnacher F. Detection of Acoustic, Electro-Optical and Radar Signatures of Small Unmanned Aerial Vehicles. *Proc. of the SPIE Security + Defence* (Edinburgh, UK, September 26–29 2016). Edinburgh, UK, 2016. P. 1–4.
6. Ezuma M., Erden F., Anjinappa C.K., Ozdemir O., Guvenc I. Micro-UAV Detection and Classification from RF Fingerprints Using Machine Learning Techniques. *Proc. of the IEEE Conference on Aerospace* (Big Sky, MT, USA, March 2–9, 2019). Big Sky, MT, USA, 2019. URL: https://scholar.google.com/scholar_lookup?journal=Proceedings+of+the+CIE+International+Conference+on+Radar+(RADAR)&title=Multi-mode+SDR+radar+platform+for+small+air-vehicle+Drone+detection&author=Y.+Kwag&author=I.+Woo&author=H.+Kwak&author=Y.+Jung&pages=1-4&.
7. Ultralytics/ultralytics. GitHub repo. URL: https://github.com/ultralytics/ultralytics.
8. Unlu E., Zenou E., Riviere N., Dupouy P.E. Deep learning-based strategies for the detection and tracking of drones using several cameras. *IPSJ Trans. Comput. Vis. Appl.* 2019. N 11. P. 7. DOI: https://doi.org/10.1186/s41074-019-0059-x.
9. Boesch G. A Guide to YOLOv8 in 2024. URL: https://viso.ai/deep-learning/yolov8-guide/.
10. VK. YoloV8 Architecture & Cow Counter With Region Based Dragging Using YoloV8. URL: https://medium.com/@VK_Venkatkumar/yolov8-architecture-cow-counter-with-region-based-dragging-using-yolov8-e75b3ac71ed8#:~:text=The%20architecture%20of%20YOLOv8%20builds,the%20backbone%20and%20the%20head.&text=A%20modified%20version%20of%20the%20CSPDarknet53%20architecture%20forms%20the%20backbone%20of%20YOLOv8.

11. Skalski P. How to Train YOLOv8 Object Detection on a Custom Dataset. URL: https://blog.roboflow.com/how-to-train-yolov8-on-a-custom-dataset/.
12. Train script. Colab. URL: https://colab.research.google.com/drive/1OwQJgBxcEHnKK3Jsp29_RgK-cHPfNjeCGw?usp=sharing#scrollTo=FBRvV5I3TMxT.
13. Confusion Matrix. WIKI. URL: https://en.wikipedia.org/wiki/Confusion_matrix.
14. Universe, webtool for test YOLOv8 models. URL: https://universe.roboflow.com/.
15. Trained model. Google Drive. URL: https://drive.google.com/file/d/14qlQScwKVd6S6GPHV7CA11oN7i1y69eU/view.

Стаття надійшла до редакції 02.04.2024