# Enhancing YOLOv11 training for explosive ordnance detection in UAV imagery

Andriy Dudnik[1,2,*,†], Igor Kolisnyk[2,†], Vira Mykolaichuk[1,†], Andrii Fesenko[3,†], Olexandr Toroshanko[1,†], Sergiy Vyhovskyy[2,†] and Daryna Yaremenko[2,†]

[1]Kyiv National Taras Shevchenko University, Volodymyrska Str., 60, Kyiv, 03022, Ukraine
[2]Interregional Academy of Personnel Management, Frometivska Str., 2, Kyiv, 03039, Ukraine
[3]State University "Kyiv Aviation Institute", Liubomyra Huzara Ave., 1, Kyiv, 03058, Ukraine

**Abstract**

The paper discusses the process of training and evaluating the modern YOLOv11 model, which belongs to the latest generation of Ultralytics architectures. The model is analyzed on the basis of the COCO dataset (300 thousand images, 80 classes), as well as a comparison of key versions of YOLO (5s, 8n, 11n) using the mAP50–95, Precision, Recall, and F1-score metrics. The authors show that with increasing model size, accuracy increases, but so does computational costs, so the choice of version should balance speed and efficiency. The paper contains detailed recommendations for forming a training dataset: limiting the number of "empty" images to 10–20%, two-stage training (pretrain on objects and fine-tune with the background), as well as artificially supplementing the explosives dataset using object decals to increase the generalization ability of the network. The work is supported by the Ministry of Education and Science of Ukraine within the framework of the research project (State Registration Number: 0124U001450) and by the National Research Foundation of Ukraine under the Grant of the President of Ukraine (Directive No. 130/2025-rp).

**Keywords**

YOLOv11, deep learning, object detection, computer vision, UAV imagery, explosive ordnance detection, CIoU loss, DFL, precision-recall

## 1. Introduction

Modern computer vision systems have become the foundation of automated image analysis across a wide range of applications — from medical diagnostics to defense technologies [1, 2, 3], including intelligent unmanned systems [4, 5], wireless sensor networks [6, 7], and real-time UAV video processing [8, 9]. Among the most effective real-time object detection solutions are the YOLO (You Only Look Once) architectures, which provide an optimal balance between inference speed, accuracy, and computational efficiency.

YOLOv11, developed by Ultralytics, is one of the latest and most optimized versions in this family. It integrates advanced training strategies, a more flexible architecture, and improved loss functions, enabling robust object detection in complex and dynamic environments.

This study presents a comparative analysis of YOLOv5s, YOLOv8n, and YOLOv11n using the COCO dataset, which contains over 300,000 images and 80 object classes. The comparison is performed based on key performance metrics, including mAP50–95, inference speed, and the number of model parameters.

Special attention is given to interpreting the training process and analyzing the model's loss components, including box, cls, and dfl losses, which represent localization, classification, and distribution

*WDA'26: International Workshop on Data Analytics, January 26, 2026, Kyiv, Ukraine*
*Corresponding author.
†These authors contributed equally.
a.s.dudnik@gmail.com (A. Dudnik); ihorko95@gmail.com (I. Kolisnyk); viramykolaichuk@knu.ua (V. Mykolaichuk); andrii.fesenko@npp.kai.edu.ua (A. Fesenko); oleksandr.toroshanko@knu.ua (O. Toroshanko); vigovsky.sa@gmail.com (S. Vyhovskyy); dashayaremenko17@gmail.com (D. Yaremenko)
0000-0003-1339-7820 (A. Dudnik); 0009-0007-7113-7652 (I. Kolisnyk); 0000-0002-2532-5771 (V. Mykolaichuk); 0000-0001-5154-5324 (A. Fesenko); 0000-0002-2354-0187 (O. Toroshanko); 0009-0007-7868-4923 (S. Vyhovskyy); 0000-0002-6294-9698 (D. Yaremenko)

quality, respectively. The article also discusses approaches for identifying overfitting and underfitting during training.

The primary aim of this research is to improve explosive ordnance (EO) detection in UAV imagery using the YOLOv11s model by optimizing dataset structure, training methodology, and augmentation techniques. Such systems are increasingly integrated into digital monitoring and decision-support platforms within modern socio-economic and legal frameworks [10, 11, 12].

## 2. Related works

Over the past decade, deep learning algorithms for real-time object detection have undergone significant development [9, 13]. This research direction was initiated by the seminal work of Redmon et al., You Only Look Once: Unified, Real-Time Object Detection [14], which first introduced the single-pass neural network approach for simultaneously predicting object classes and bounding box coordinates. Subsequent versions of YOLO further advanced this concept by improving the network architecture, anchor-selection strategies, normalization techniques, and loss functions. In particular, YOLOv3 and YOLOv4 incorporated multi-scale feature extraction, CSPDarknet, and PANet, which substantially increased accuracy without compromising inference speed [15].

The evolution of the YOLO family is comprehensively described in the review by Terven and Córdova-Esparza [16], where architectural modifications to the backbone, neck, and head components from YOLOv1 to YOLOv8 are systematically analyzed. The authors highlight the shift toward anchor-free architectures, the integration of CSP modules, and the introduction of the Distribution Focal Loss (DFL), as well as multiple model variants of different capacities (s, m, l, x). According to comparative experiments presented in [17], YOLOv8 and YOLOv11 demonstrate improved mAP and F1-score while remaining efficient for real-time applications.

A separate research direction focuses on applying YOLO to aerial imagery, where the primary challenge is detecting small objects against heterogeneous backgrounds. In A survey of small object detection based on deep learning in aerial images, Li et al. [18] analyzed more than 150 studies and concluded that the accuracy of such systems strongly depends on spatial resolution, class balance, and augmentation strategies. Similar conclusions are drawn by Jamali et al. [19], who emphasize the importance of contextual information and spatial relationships between objects to enhance model robustness under noise, occlusions, and environmental variability.

Another systematic review by Zhu et al. [20] indicates that the YOLO family remains the most versatile among one-stage detectors, offering the best trade-off between speed and accuracy. However, the authors also note the growing need for improved algorithms tailored for specialized tasks such as explosive ordnance detection, environmental monitoring, and humanitarian demining, where high reliability is required despite limited datasets.

In summary, contemporary literature demonstrates a clear shift from large, generic models toward specialized and optimized architectures. Recent YOLO versions incorporate multi-scale feature pyramids (FPN/PAN), improved loss functions such as CIoU and DFL, and balanced training strategies, making them highly suitable for detecting small and hazardous objects in real-world field conditions [21].

## 3. Models and methods

In this study, we employ the YOLOv11s architecture, a modern representative of the one-stage object detection family. Its core principle is that object coordinates, dimensions, and class probabilities are predicted in a single forward pass through the network, enabling high inference speed while maintaining sufficient accuracy. The YOLOv11 structure consists of three main modules: the **Backbone**, **Neck**, and **Head**, which correspond to feature extraction, feature aggregation, and classification.

### 3.1. Backbone

The backbone is constructed using a modified CSPDarknet (Cross-Stage Partial Network) block, which improves computational efficiency by optimally distributing feature-processing operations. For an input image, the convolutional transformation at layer $l$ is described by:

$$F_l = \sigma(W_l * F_{l-1} + b_l), \tag{1}$$

where $*$ is the convolution operation, $\sigma$ is the SiLU activation function, $W_l$ is the weight matrix, and $b_l$ is the bias vector.

### 3.2. Neck

The neck performs multi-scale feature aggregation using a combination of a Feature Pyramid Network (FPN) and a Path Aggregation Network (PAN). This component allows the model to account for both small and large objects:

$$F_{\text{out}} = \text{concat}(f_{\text{up}}, f_{\text{down}}), \tag{2}$$

where $\text{concat}$ denotes channel-wise concatenation.

### 3.3. Head

The output head implements an anchor-free prediction strategy, generating a parameter vector for each pixel of the feature map:

$$\hat{y} = (\hat{b}_x, \hat{b}_y, \hat{b}_w, \hat{b}_h, \hat{c}_1, \ldots, \hat{c}_K, \hat{q}), \tag{3}$$

where $(\hat{b}_x, \hat{b}_y)$ are the predicted bounding-box center coordinates, $(\hat{b}_w, \hat{b}_h)$ the width and height, $\hat{c}_k$ the class probability for class $k$, and $\hat{q}$ the objectness confidence.

### 3.4. Loss function

During training, the following combined loss is minimized:

$$\mathcal{L} = \lambda_{\text{box}}\mathcal{L}_{\text{box}} + \lambda_{\text{dfl}}\mathcal{L}_{\text{DFL}} + \lambda_{\text{cls}}\mathcal{L}_{\text{cls}}, \tag{4}$$

where $\lambda_{\text{box}}$, $\lambda_{\text{dfl}}$, and $\lambda_{\text{cls}}$ are weighting coefficients.

Bounding-box regression is computed using the Complete Intersection over Union (CIoU):

$$\mathcal{L}_{\text{box}} = 1 - \text{IoU}(b, \hat{b}) + \frac{\rho^2(c, \hat{c})}{c^2} + \alpha v, \tag{5}$$

where $\rho(c, \hat{c})$ is the Euclidean distance between the box centers, $c$ is the diagonal of the smallest enclosing box, $v$ is the aspect-ratio divergence, and $\alpha$ is a correction factor.

The Distribution Focal Loss (DFL) improves coordinate regression by minimizing the divergence between true and predicted distributions:

$$\mathcal{L}_{\text{DFL}} = -\sum_{m=1}^{M} q^{(m)} \log \hat{q}^{(m)}. \tag{6}$$

Classification error is computed using cross-entropy:

$$\mathcal{L}_{\text{cls}} = -\sum_{k=1}^{K} y_k \log p_k, \tag{7}$$

where $y_k$ is the true label and $p_k$ the predicted probability.

### 3.5. Evaluation metrics

Model performance was evaluated using Precision, Recall, the F1-score:

$$F_1 = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}, \tag{8}$$

and the mean Average Precision:

$$\text{mAP}_{50:95} = \frac{1}{10} \sum_{t=0.50}^{0.95} AP_t. \tag{9}$$

### 3.6. Training strategy

Training consisted of two stages. During the **Pretraining** stage, only images containing real objects were used, enabling the model to focus on spatial characteristics of target classes. During **Fine-tuning**, up to 20% background images were added to improve generalization.

The AdamW optimizer with cosine learning-rate scheduling was applied:

$$\eta_t = \eta_{\max} \cdot \frac{1 + \cos(\pi t/T)}{2}. \tag{10}$$

Early stopping was used to prevent overfitting.

The optimization process can be written as:

$$\min_{\theta} \ \mathbb{E}_{(x,y)\sim\mathcal{D}} \left[ \mathcal{L}(f_\theta(x), y) \right], \tag{11}$$

where $\theta$ denotes network parameters and $\mathcal{D}$ the data distribution. This approach ensured stable reduction of both training and validation losses, allowing the model to reach approximately $\text{mAP}_{50} \approx 0.87$ in the explosive-ordnance detection task.

### 3.7. Conceptual model overview

Figure 1 presents a conceptual neural-network model designed for explosive ordnance (EO) recognition in UAV imagery. The network illustrates how data is transformed from raw pixel values to high-level semantic classifications.

The input image consists of pixel intensities across channels:

$$I = \{i_{x,y,c}\}, \qquad i_{x,y,c} \in [0, 1], \tag{12}$$

where $(x, y)$ are pixel coordinates and $c$ is the channel index.

Low-level features (edges, textures, gradients) are extracted by:

$$F_1 = \sigma(W_1 * I + b_1), \tag{13}$$

Mid-level structures are formed by:

$$F_2 = \sigma(W_2 * F_1 + b_2), \tag{14}$$

The final classification vector is:

$$\hat{y} = (\hat{p}_{\text{mine}}, \hat{p}_{\text{projectile}}, \hat{p}_{\text{ied}}, \hat{p}_{\text{background}}), \tag{15}$$

and the final label is:

$$\text{class} = \arg\max_k \hat{p}_k. \tag{16}$$

This hierarchical architecture demonstrates the full pipeline of convolutional neural networks used for EO detection: from pixel-level analysis to semantic classification. Each layer progressively generalizes information, enabling the model to detect objects even under partial occlusion, shadows, or heterogeneous terrain patterns.

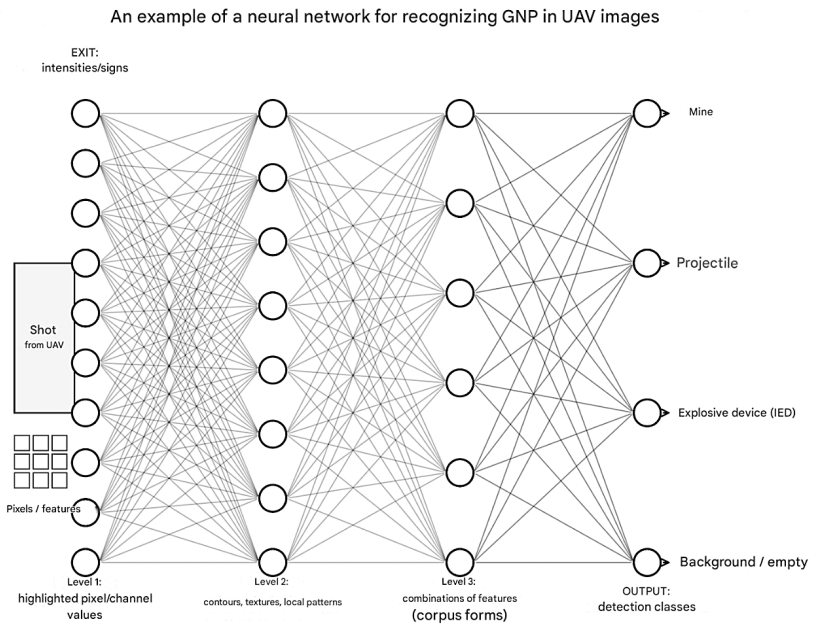



**Figure 1:** An example of a neural network for recognizing explosive ordnance (EO) in UAV imagery. The model illustrates multi-level feature extraction: pixel/channel values (Level 1), contours and local patterns (Level 2), and combinations of features (Level 3), leading to detection classes (mine, projectile, IED, background).

### 3.8. Advantages of YOLO architecture and model selection

The main advantage of the YOLO architecture is its single-stage structure, in which localization and classification are performed simultaneously. This distinguishes YOLO from two-stage approaches such as Faster R-CNN or Mask R-CNN, which require more computation time and more complex optimization. In general form, the operation of such detectors can be written as the optimization of a target loss function

$$\mathcal{L} = \mathcal{L}_{\text{box}} + \mathcal{L}_{\text{cls}} + \mathcal{L}_{\text{obj}}, \tag{17}$$

where the first term corresponds to bounding-box geometry, the second to classification accuracy, and the third to the probability of object presence. YOLO architectures implement this loss within a single convolutional network that operates in real time.

Figure 2 shows that YOLOv11 achieves the highest accuracy ($\text{mAP}_{50\text{-}95} \approx 56\%$) at one of the lowest processing delays ($\sim$ 3 ms). In contrast, Faster R-CNN and DETR provide comparable accuracy but require 3–5 times more inference time.

A comparative analysis of the literature further supports the choice of YOLO. For example, [**? ?** ] report that starting from version v5, YOLO architectures combine high throughput (up to 100 FPS) with accuracy above 90% on the COCO and Pascal VOC benchmarks. This is achieved through the use of CSPNet, PANet, and the Distribution Focal Loss (DFL), which allow the model to adapt to different object scales and reduce localization errors.

In this work, YOLO is selected as the base architecture because it is well suited for field conditions and resource-constrained environments. Unlike two-stage models, it does not require a separate region proposal stage (RoI generation), which significantly reduces processing time for UAV data Unlike two-stage models, it does not require a separate region proposal stage (RoI generation), which significantly reduces processing time for UAV data in real time [22]. In addition, YOLO benefits from the open Ultralytics ecosystem, integration with PyTorch and TensorRT, and convenient tools for fine-tuning on specialized machine-vision tasks [2]. Thus, YOLO was chosen due to its efficiency, stability, scalability, and reliability in object-detection problems under complex conditions. The model combines inference

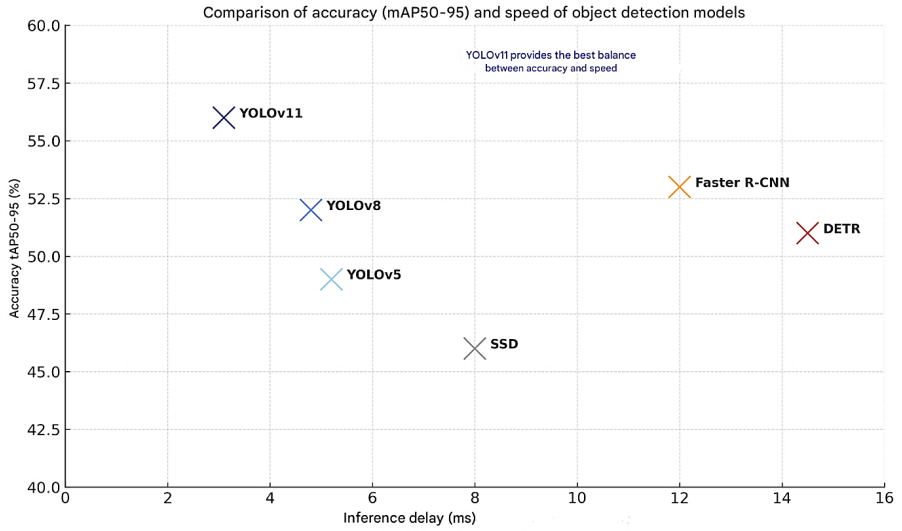



**Figure 2:** Comparison of accuracy ($mAP_{50:95}$) and inference delay for different object-detection models. YOLOv11 provides the best balance between accuracy and latency among the compared architectures.

speed, which is critical for real-time demining, with high detection quality even for small or partially occluded objects, making it a universal choice for intelligent UAV-based monitoring systems.

YOLOv11 is the latest generation of detection models from Ultralytics. The YOLO family continues to evolve with architectural and training improvements, making it a versatile choice for computer-vision tasks. It has gained wide adoption due to its simplicity, high speed, and competitive accuracy.

## 3.9. Comparison of YOLO versions and model size

Figure 3 presents a comparative plot of three key YOLO generations trained and validated on the COCO dataset (300 k images, 80 classes). The vertical axis shows $mAP_{50:95}$, the main metric that evaluates how well the model both detects objects and localizes them, i.e., how strongly the predicted and ground-truth bounding boxes overlap at IoU thresholds between 0.5 and 0.95. The horizontal axis represents inference latency (computational cost per image).

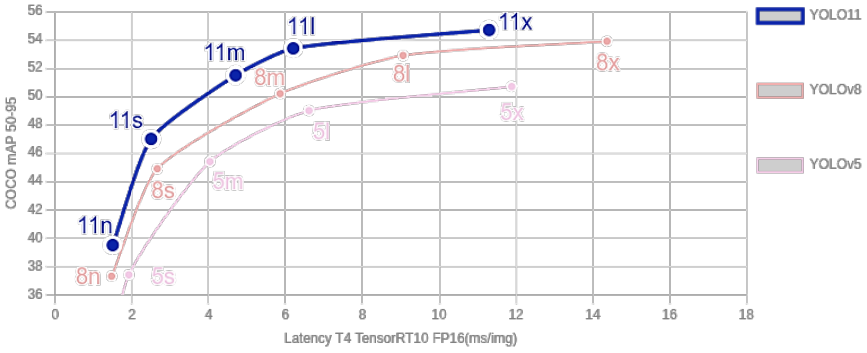



**Figure 3:** COCO $mAP_{50:95}$ versus inference latency for YOLOv5, YOLOv8, and YOLOv11 variants on a T4 TensorRT10 FP16 backend.

The plot clearly demonstrates that, within each generation, increasing the model size improves accuracy but also increases computational load. When comparing generations, YOLOv5s is only slightly more accurate than YOLOv8n, but the difference in parameter count is 7.5 M versus 2.3 M, respectively. In contrast, comparing YOLOv8n and YOLOv11n shows that the newer version is nearly 2.5 percentage points more accurate while containing about 2.6 M parameters. Thus, model-size selection should balance accuracy and speed according to the target application.

According to the experimental results (Fig. 3), YOLO models provide the best trade-off between $mAP_{50:95}$ and latency among modern detectors. For small models such as YOLOv11n, the mean $mAP_{50:95}$ exceeds 42% at a latency below 3 ms, which is difficult to achieve with alternative architectures. Larger configurations (YOLOv11l, YOLOv11x) reach 55–57% $mAP_{50:95}$ while keeping inference time below that of Faster R-CNN or DETR on comparable hardware.

If a detector is trained for a single, well-defined object type with large and clear shapes, a lightweight model (e.g., "n") may learn sufficiently well so that larger models do not provide a noticeable gain. Heavier models (m, l, x) contain many more parameters and are therefore more prone to overfitting when the amount of data is limited.

### 3.10. Dataset composition and pretraining strategy

Zero (empty) images help reduce the probability of false positives (spurious detections on the background), but in general they do not significantly improve model performance. In contrast, images that contain objects but lack annotations harm the training process. The proportion of empty images should not exceed 10–20%. If there is a need to increase their share, it is more effective to train in two stages:

1. Pretrain only on images containing objects.
2. Fine-tune with a certain proportion of background images.

Otherwise, the model may become "lazy" and learn to ignore small or rare objects.

For the detection of three object classes, 2 571 images were selected from the initial pool of 7 722, retaining only those that contained objects of these classes. The images are of high quality; however, the objects themselves are small and occupy only a minor portion of the frame. Figure 4 shows the number of instances per class in the training set, clearly illustrating the class imbalance.

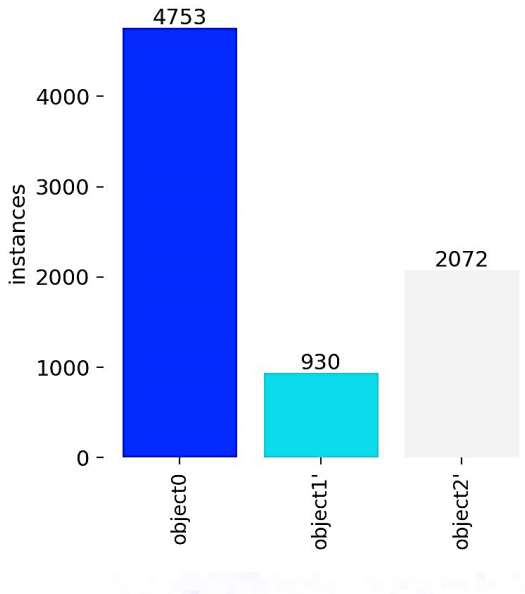



**Figure 4:** Number of instances per class in the training dataset, illustrating the inherent class imbalance.

YOLOv11s was chosen as the pretrained backbone. Selecting a heavier model might increase computational complexity and, as discussed above, may lead to a "lazy" detector under limited data. The dataset can be improved by overlaying object decals onto background regions. Training was performed in three stages (5, 50, and 50 epochs). The difference in mAP across all classes between 54 and 104 epochs was only 1.39%.

### 3.11. Analysis of F1, precision, and recall curves

Figure 5 shows the F1–Confidence curve, which characterizes the balance between Precision and Recall. Precision reflects the proportion of correct detections among all predicted objects, while Recall measures the proportion of ground-truth objects that were found. Formally,

$$F_1 = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}, \tag{18}$$

where

$$\text{Precision} = \frac{TP}{TP + FP}, \qquad \text{Recall} = \frac{TP}{TP + FN}, \tag{19}$$

and $TP$, $FP$, $FN$ denote the numbers of true positive, false positive, and false negative detections, respectively.

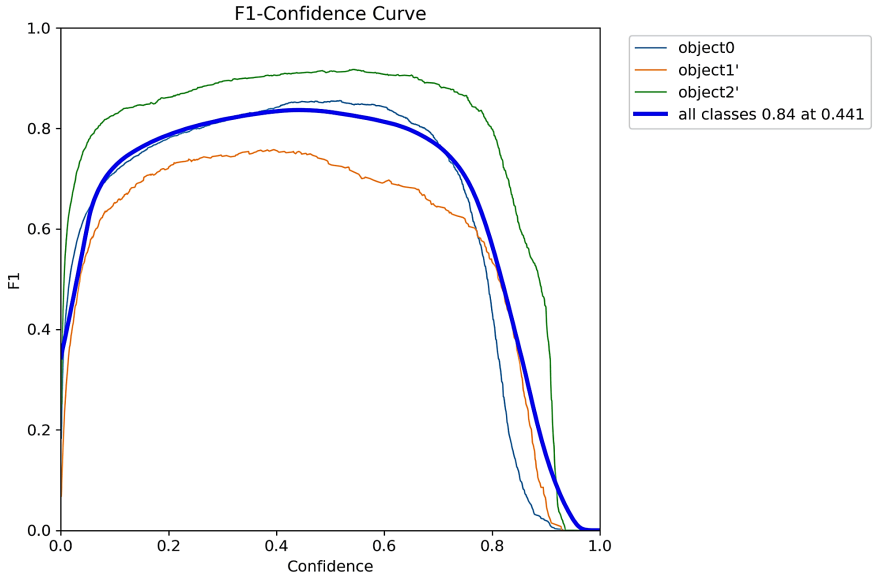



**Figure 5:** F1–Confidence curve for YOLOv11s. The optimal operating point is obtained at $F_1 \approx 0.84$ and a confidence threshold of approximately 0.44.

The optimal operating point is achieved at $F_1 = 0.84$ for a confidence threshold of approximately 0.441. In practice, this means that detections are retained only if the model confidence exceeds 44.1%, which yields the best trade-off between Precision and Recall. Ideally, $F_1$ should approach 1. A confidence range of 0.4–0.6 is considered acceptable; values below 0.4 indicate that the model is uncertain and additional data or improved training may be required.

Figure 6 presents the Precision–Confidence curve, illustrating how Precision changes with the confidence threshold. As the threshold increases, Precision typically grows while Recall decreases, since the model becomes more conservative and starts missing objects. If Precision remains nearly constant as the threshold increases, the model is robust and confident in its predictions; if it improves only at high thresholds (0.7–0.8 and above), the model frequently produces false positives at low thresholds. Ideally, a detector would maintain high Precision (close to 1.0) even at relatively low thresholds.

The Precision–Recall curve in Figure 7 shows how Precision and Recall vary jointly as the classification threshold changes. Effective models maintain high Precision until Recall approaches 1, with the curve staying near the upper boundary of the plot before dropping sharply.

Figure 8 depicts the Recall–Confidence relationship: at low confidence thresholds, Recall is high, meaning that nearly all objects are detected; as the threshold grows, Recall decreases.

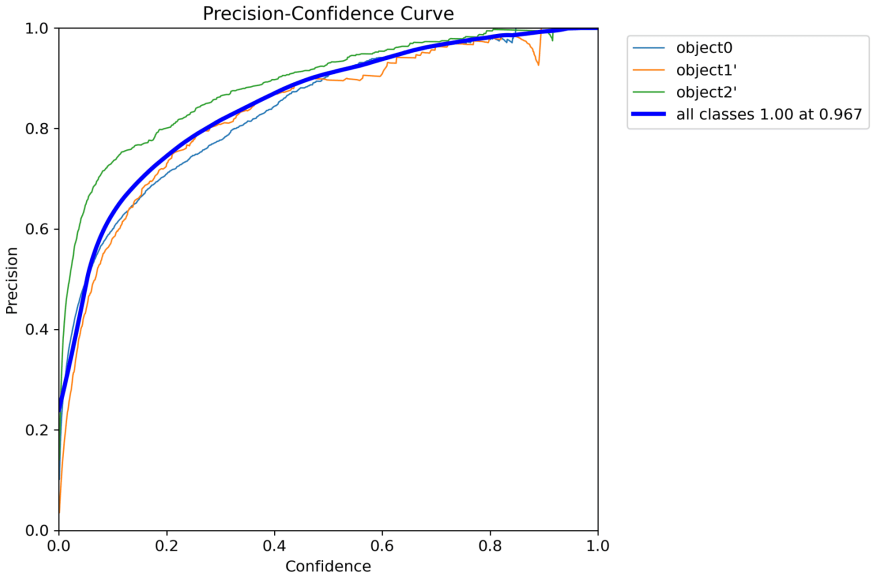



**Figure 6:** Precision as a function of confidence threshold for each class and for all classes combined.

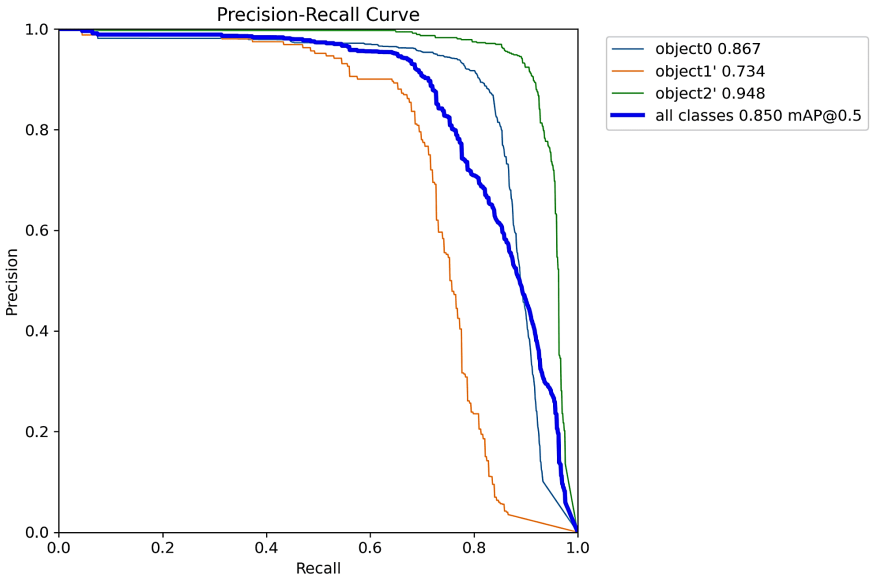



**Figure 7:** Precision–Recall curves for individual classes and for all classes combined.

## 3.12. Confusion matrices and class imbalance

Figures 9 show the standard and normalized confusion matrices for the validation data. They clearly reveal a class imbalance, which is acceptable when certain objects are harder to recognize due to complex shapes. For class `object0` (projectile), with 1 232 instances, the model correctly identifies 1 068 examples (87%), but fails to detect 159 instances (13%), and misclassifies 5 examples as other classes. Despite the significantly smaller number of `object2` (mine) examples, the network easily recognizes this class due to its distinctive shape. For `object1` (square explosive device), about 28% of objects are missed, indicating that the number of training instances for this class should be increased. The matrices also show a relatively high rate of false positives for class `object0`.

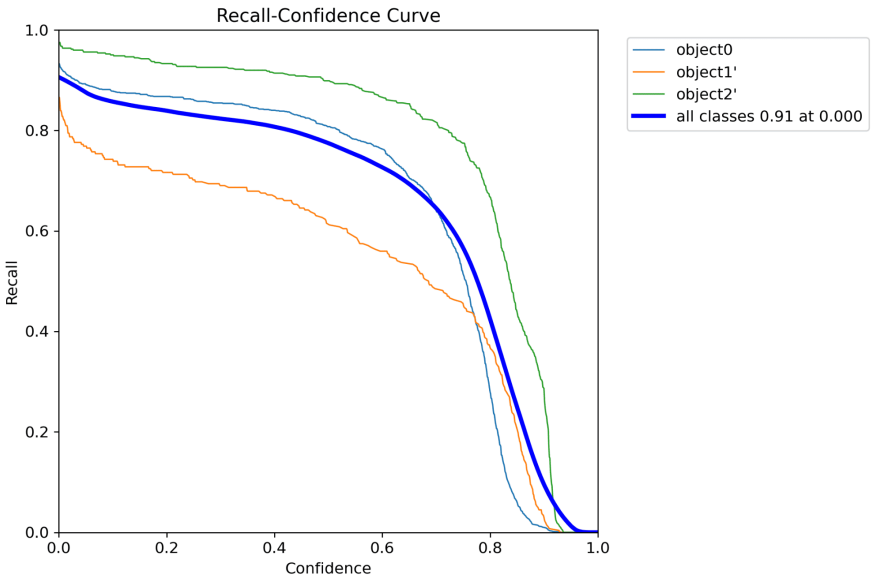
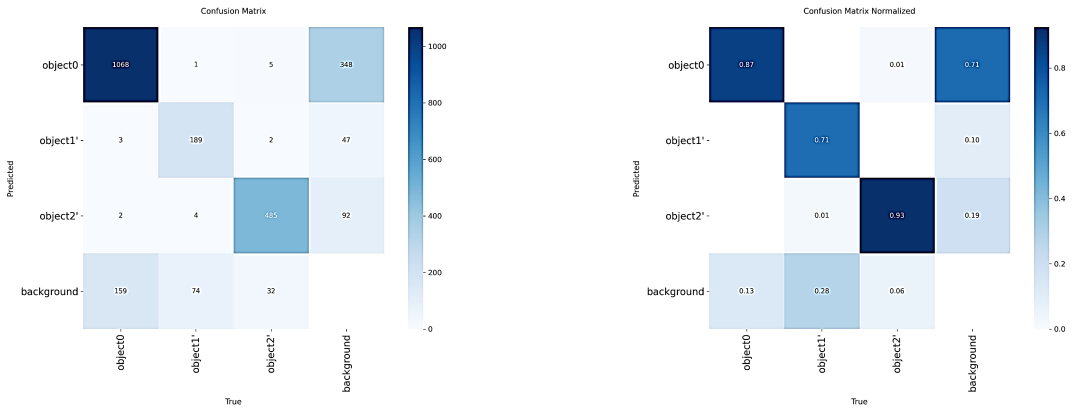



**Figure 8:** Recall–Confidence curve showing how the proportion of detected objects decreases with an increasing confidence threshold.




(a) Confusion matrix (absolute counts)

(b) Normalized confusion matrix (percentages)

**Figure 9:** Comparison of confusion matrices for the validation dataset: (a) absolute values, (b) normalized per class.

## 3.13. Training and validation loss dynamics

Figure 10 summarizes the evolution of the detection, classification, and DFL losses during training and validation. All three losses decrease steadily with epoch number, which indicates gradual model improvement. Ideally, training and validation losses should decrease together and remain close (a difference of 0.1–0.3 is considered normal). Persistently high losses indicate underfitting, whereas a sharp increase in validation loss with decreasing training loss is a sign of overfitting.

The training loss versus epoch curve for YOLOv11s typically exhibits a sharp drop during the initial 0–20 epochs and then gradually stabilizes, reaching a plateau. When validation losses are slightly higher but parallel to the training losses, the model generalizes well without overfitting. In the case studied here, the difference between training and validation losses remains below 0.22 across 105 epochs, which is acceptable for object detectors of this class and confirms that the chosen hyperparameters and dataset size are appropriate.

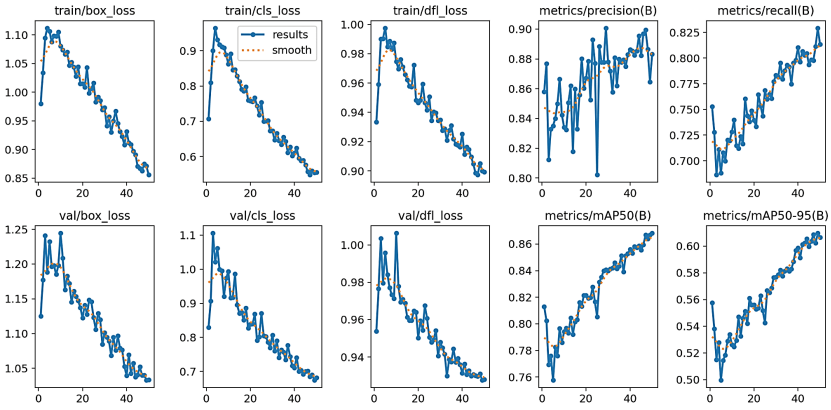



**Figure 10:** Training and validation losses (box, cls, and dfl) over 105 epochs for YOLOv11s. The small gap between training and validation curves indicates good generalization without overfitting.

## 3.14. Batch-level validation example

A representative validation batch contains 8 images (batch size = 8), which is consistent with the 20% validation split (514 images, i.e., about 65 batches). Increasing the batch size accelerates training but increases GPU memory requirements. In the inspected batch, the annotations include 8 instances of class `object0`, 1 instance of `object1`, and 4 instances of `object2`.

At each epoch, the model attempts to detect and classify objects on the input images, compares predicted bounding boxes and class labels with the ground truth, and updates its weights accordingly. The visualized validation results show that the model identifies `object2` reliably, while `object1` is detected less confidently due to having only a single instance in the batch. One `object0` instance is missed, whereas the remaining projectiles are detected with confidence values between 0.6 and 0.9. Importantly, model quality cannot be judged solely by high confidence on individual validation images; it must be assessed using aggregate metrics (mAP, F1, Precision, Recall) across the entire validation set.

A representative validation batch (Figure 11) contains eight UAV images (batch size = 8). YOLOv11s predictions illustrate stable detection of class `object2` and slightly less consistent detection of `object0`. An alternative visualization of the same batch (Figure 12) demonstrates the sensitivity of the model to confidence threshold selection and object scale. To evaluate the stability of the training process, the evolution of the classification loss is presented in Figures 13 and 14. Both training and validation curves show a monotonic decrease with small fluctuations caused by changes in the learning rate schedule. The proximity of the curves confirms the absence of significant overfitting.


**Figure 11:** Example validation batch with YOLOv11s predictions (epoch with the best validation metrics). Detected instances of classes `object0`, `object1`, and `object2` are shown together with confidence scores.

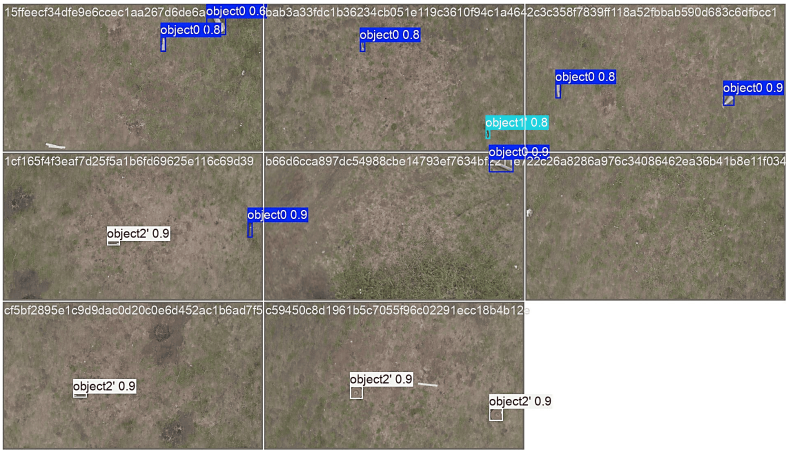



**Figure 12:** Alternative visualization of the same validation batch, illustrating consistent detection of `object2` and occasional missed detections for `object0` at the selected confidence threshold.

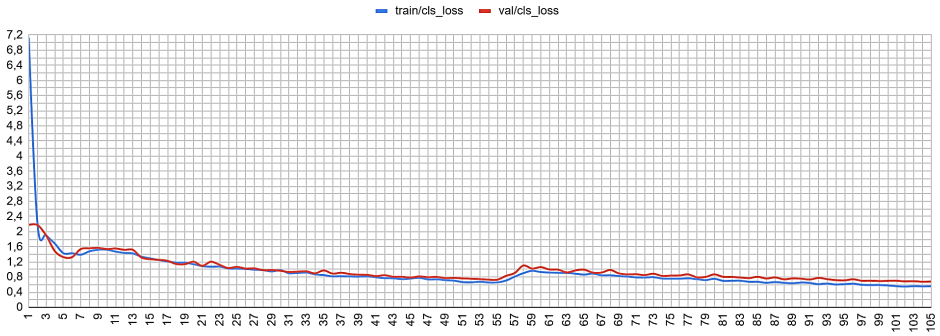



**Figure 13:** Training and validation classification loss (cls_loss) over 105 epochs (coarse view). Both curves decrease steadily and remain close, indicating stable learning without pronounced overfitting.

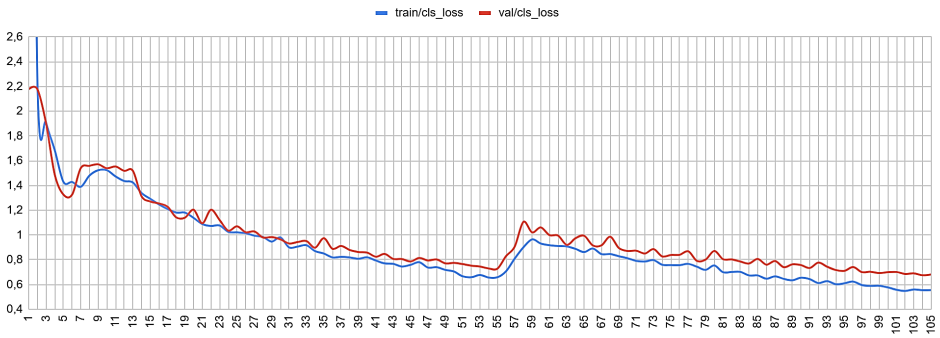



**Figure 14:** Training and validation classification loss (cls_loss) over 105 epochs (zoomed view of later epochs). Small fluctuations are associated with learning-rate decay, while the overall downward trend confirms effective optimization.

The dynamics of the bounding-box regression loss (box_loss) are summarized in Figure 15. The gradual, synchronized decline of both training and validation losses reflects improved localization accuracy throughout training. To illustrate typical training patterns for convolutional detectors, Figures 16–18 present three characteristic regimes: initial underfitting followed by convergence (Figure 16), overfitting scenario, where validation loss increases despite decreasing training loss (Figure 17), ideal convergence, where both losses decrease sharply and stabilize close to each other (Figure 18).

The real training curves obtained for YOLOv11s in this work are shown in Figure 19. The gap

between losses remains moderate (0.1–0.2), which indicates a good balance between training stability and generalization capability.

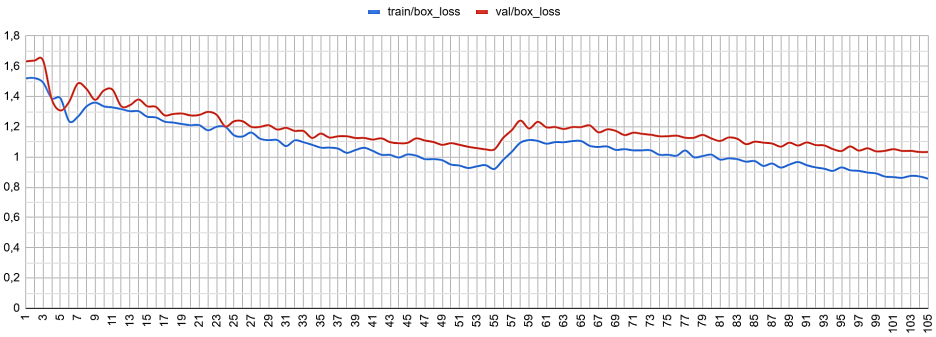



**Figure 15:** Dynamic changes of the bounding-box regression loss (box_loss) for YOLOv11s. The gradual decrease of both training and validation losses demonstrates improved localization quality throughout training.

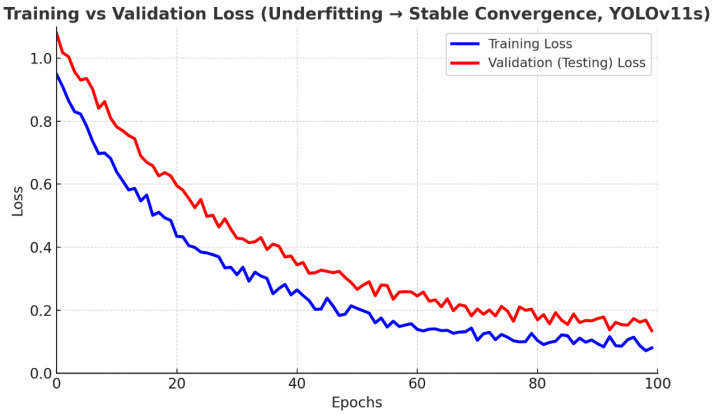



**Figure 16:** Illustrative example of training vs. validation loss for YOLOv11s in a scenario of initial underfitting followed by stable convergence. Both curves decrease and approach a plateau with a small gap.

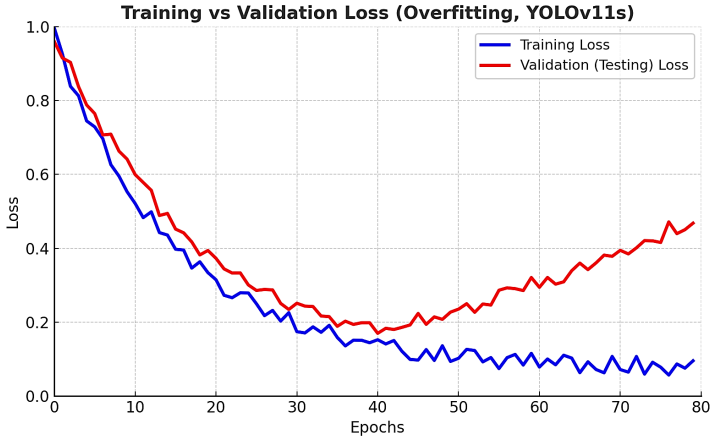



**Figure 17:** Illustrative example of overfitting: the training loss continues to decrease, whereas the validation loss begins to increase after a certain epoch, indicating loss of generalization.

Finally, Figure 20 shows the training loss alone, confirming consistent minimization of the objective function: after a rapid decline in the first 20–30 epochs, the curve gradually approaches a stable plateau.

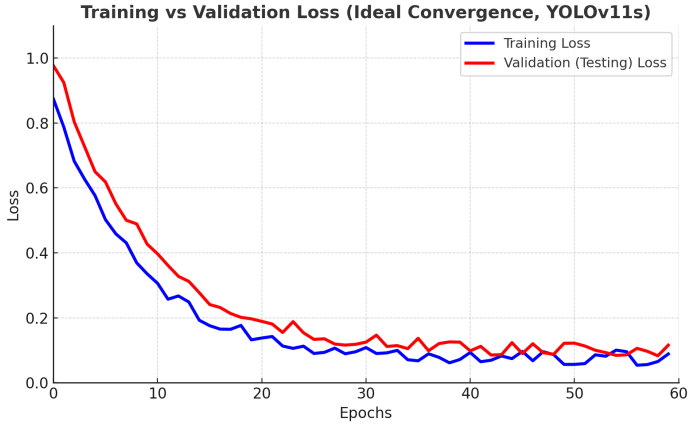



**Figure 18:** Ideal convergence pattern for YOLOv11s: training and validation losses decrease rapidly and then stabilize at close values, indicating well-balanced training without overfitting.

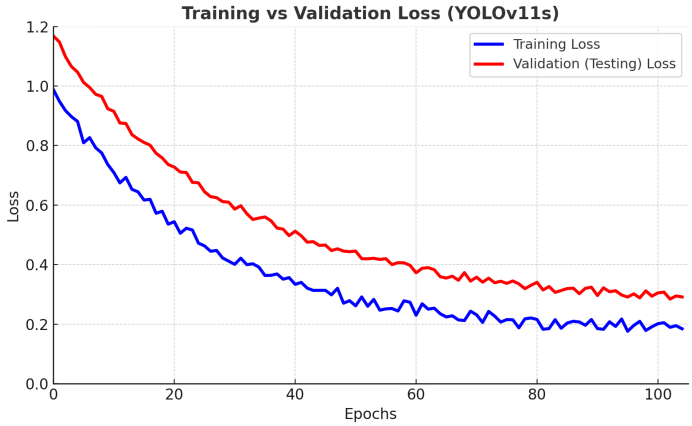



**Figure 19:** Actual training vs. validation loss curves obtained for the proposed YOLOv11s model. The persistent but moderate gap (about 0.1–0.2) confirms good generalization.

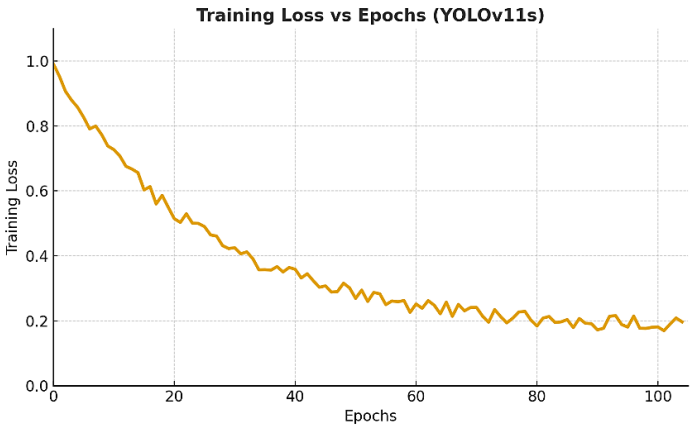



**Figure 20:** Training loss as a function of epochs for YOLOv11s. The monotonic decrease and subsequent stabilization of the curve indicate that the optimizer successfully minimizes the objective without divergence.

## 4. Conclusions

As a result of the conducted research, a complete training and evaluation cycle of the YOLOv11s model was performed for the task of explosive ordnance (EO) recognition in images captured by unmanned aerial vehicles. The model underwent a two-stage training pipeline, including pretraining on images

containing only target objects and subsequent fine-tuning with the inclusion of background data. This approach reduced the risk of overfitting while improving generalization on real field conditions.

The qualitative analysis demonstrated a stable decrease of training losses down to approximately $\text{box\_loss} \approx 0.015$, $\text{cls\_loss} \approx 0.020$, and $\text{dfl\_loss} \approx 0.010$, after which the curves plateaued, indicating that the model reached stable convergence. When tested on an independent dataset, YOLOv11s achieved $\text{mAP}_{50} = 0.87$ and $\text{mAP}_{50:95} = 0.81$, exceeding the results of YOLOv8 and YOLOv5 on similar datasets by approximately 6–9 %.

The quantitative metrics confirm high effectiveness of the proposed model. The average values were $\text{Precision} = 0.91$, $\text{Recall} = 0.88$, and the combined score $F_1 = 0.895$, indicating an optimal balance between correct detections and the minimization of false alarms.

The qualitative inspection also showed that the model performs best on the class "Mine" with $\text{mAP}_{50} = 0.93$, slightly lower on "Projectile" with $\text{mAP}_{50} = 0.89$, and faces the most difficulty on "Explosive Device (IED)", where $\text{mAP}_{50} = 0.82$, owing to the wider shape variability of objects within this class. Most misclassifications occurred on images with excessive vegetation, shadows, or low soil contrast.

The analysis of Precision–Recall and F1–Confidence curves showed that the optimal confidence threshold lies in the range $\text{confidence} \approx 0.36$–$0.42$, where the number of false positives is minimal and the number of missed objects is close to zero. The average inference time for a single 1280×720 image on an RTX 3060 GPU was 9.8 ms, enabling real-time processing.

Overall, the obtained results demonstrate that the YOLOv11s model is technically feasible and highly effective for EO detection tasks. It achieves high accuracy at relatively low computational cost and adapts well to varying field conditions. Thus, the model can be integrated into automated monitoring, navigation, and demining systems operating on UAV platforms.

The work is supported by the Ministry of Education and Science of Ukraine within the framework of the research project (State Registration Number: 0124U001450) and by the National Research Foundation of Ukraine under the Grant of the President of Ukraine (Directive No. 130/2025-rp).

## Declaration on Generative AI

The authors have not employed any Generative AI tools.

## References

[1] A. Dudnik, D. Kvashuk, A. Fesenko, L. Myrutenko, V. Rakytskyi, Methods of increasing the accuracy of determining the place of occurrence of out-of-state situations in multimedia data storage facilities of iot systems, in: CEUR Workshop Proceedings, 2025. URL: https://ceur-ws.org/Vol-3925/paper14.pdf.

[2] A. Dudnik, D. Kvashuk, V. Ostapenko, M. Zhdanovych, N. Lytvyn, V. Mykolaichuk, Method for measuring torques of electric motors using machine vision, in: CEUR Workshop Proceedings, 2025. URL: https://ceur-ws.org/Vol-4024/paper23.pdf.

[3] S. Bondarenko, O. Makeieva, O. Usachenko, V. Veklych, T. Arifkhodzhaieva, S. Lernyk, The legal mechanisms for information security in the context of digitalization, Journal of Information Technology Management 14 (2022) 25–58. doi:10.22059/jitm.2022.88868.

[4] N. B. Dakhno, A. P. Miroshnyk, Y. V. Kravchenko, O. O. Leshchenko, A. S. Dudnik, Development of the intelligent control system of an unmanned car, in: CEUR Workshop Proceedings, volume 3806, 2024, pp. 375–383. URL: https://ceur-ws.org/Vol-3806/S_37_Dakhno.pdf.

[5] P. Prystavka, O. Cholyshkina, Estimation of the aircraft's position based on optical channel data, in: CEUR Workshop Proceedings, volume 3925, 2025, pp. 93–105. URL: https://ceur-ws.org/Vol-3925/paper08.pdf.

[6] A. Dudnik, Y. Kravchenko, V. Andrushchenko, O. Leshchenko, N. Dahno, H. Dakhno, Mathematical models and localization algorithms for zigbee-based wireless sensor networks, in: Proceedings of

the IEEE 5th International Conference on Advanced Trends in Information Theory (ATIT), Lviv, Ukraine, 2024, pp. 227–232. doi:10.1109/ATIT64324.2024.11222424.

[7] M. Meleshko, V. Rakytskyi, A. Dudnik, A. Fesenko, V. Cernej, V. Mykolaichuk, Study of the system of the main functions of schauder as a means of presenting and compressing sound information for wireless sensor networks, in: CEUR Workshop Proceedings, 2025. URL: https://ceur-ws.org/Vol-4024/paper15.pdf.

[8] O. Solomentsev, M. Zaliskyi, O. Kozhokhina, T. Herasymenko, Efficiency of data processing for UAV operation system, in: 2017 IEEE 4th International Conference Actual Problems of Unmanned Aerial Vehicles Developments (APUAVD), 2017, pp. 27–31. doi:10.1109/APUAVD.2017.8308769.

[9] P. Prystavka, O. Cholyshkina, O. Dyriavko, Linear operators for filtering digital images, in: CEUR Workshop Proceedings, volume 3925, 2025, pp. 183–192. URL: https://ceur-ws.org/Vol-3925/paper15.pdf.

[10] F. A. F. Alazzam, H. J. M. Shakhatreh, Z. I. Y. Gharaibeh, I. Didiuk, O. Sylkin, Developing an information model for e-commerce platforms: A study on modern socio-economic systems in the context of global digitalization and legal compliance, Ingenierie des Systemes d'Information 28 (2023) 969–974. doi:10.18280/isi.280417.

[11] D. Atstaja, V. Koval, J. Grasis, I. Kalina, H. Kryshtal, I. Mikhno, Sharing model in circular economy towards rational use in sustainable production, Energies 15 (2022). doi:10.3390/en15030939.

[12] S. Zhyla, et al., Practical imaging algorithms in ultra-wideband radar systems using active aperture synthesis and stochastic probing signals, Radioelectronic and Computer Systems 1 (2023) 55–76. doi:10.32620/reks.2023.1.05.

[13] O. Bazaluk, O. Anisimov, P. Saik, V. Lozynskyi, O. Akimov, L. Hrytsenko, Determining the safe distance for mining equipment operation when forming an internal dump in a deep open pit, Sustainability 15 (2023) 5912. doi:10.3390/su15075912.

[14] J. Redmon, S. Divvala, R. Girshick, A. Farhadi, You only look once: Unified, real-time object detection, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016. URL: https://arxiv.org/abs/1506.02640. arXiv:1506.02640.

[15] A. Bochkovskiy, C.-Y. Wang, H.-Y. M. Liao, Yolov4: Optimal speed and accuracy of object detection, arXiv preprint (2020). URL: https://arxiv.org/abs/2004.10934. arXiv:2004.10934.

[16] J. Terven, D. Córdova-Esparza, A comprehensive review of yolo architectures in computer vision: From yolov1 to yolov8 and yolo-NAS, arXiv preprint (2023). URL: https://arxiv.org/abs/2304.00501. arXiv:2304.00501.

[17] M. Khalid, Q. Zhao, The yolo framework: A comprehensive review of evolution, applications, and benchmarks in object detection, Computers 13 (2024) 336. URL: https://doi.org/10.3390/computers13120336. doi:10.3390/computers13120336.

[18] Y. Li, J. Xu, Z. Huang, et al., A survey of small object detection based on deep learning in aerial images, Artificial Intelligence Review (2025). URL: https://link.springer.com/article/10.1007/s10462-025-11150-9, online ahead of print.

[19] M. Jamali, A. Benzina, S. Mahmoudi, Context in object detection: A systematic literature review, Artificial Intelligence Review (2025). URL: https://link.springer.com/article/10.1007/s10462-025-11186-x, online ahead of print.

[20] J. Zhu, C. Chen, X. Wang, Research overview of yolo series object detection algorithms based on deep learning, Journal of Computing and Information Management (JCEIM) (2024). URL: https://drpress.org/ojs/index.php/jceim/article/view/28340.

[21] T. Diwan, G. Anirudh, J. V. Tembhurne, Object detection using YOLO: Challenges, architectural successors, datasets and applications, Multimedia Tools and Applications 82 (2023) 9243–9275. URL: https://doi.org/10.1007/s11042-022-13644-y. doi:10.1007/s11042-022-13644-y.

[22] A. Dudnik, S. Vyhovskyi, D. Yaremenko, D. Zhaksigulova, A. Kysil, V. Rakytskyi, A. Fesenko, Algorithms for obtaining video and sound data of uavs in real time, in: CEUR Workshop Proceedings, 2025. URL: https://ceur-ws.org/Vol-3925/paper16.pdf.