**International Journal of Emerging Engineering and Technology (IJEET)**
ISSN (e): 2958-3764
Volume: 5, Number: 1-1, Pages: 51- 59, Year: 2026

# YOLOv8-P2-CBAM-NWD: High-Precision Small-Target Drone Detection in Complex Visual Airspace via Shallow Attention Routing and Normalized Wasserstein Distance

Layeq Ali [1,*], Iftikhar Ahmed Khan [2] and Shahid Yousaf [2]

[1] Department of Intelligent System, University of Lahore, Lahore, Pakistan
[2] Department of Computer Science, University of Lahore, Lahore, Pakistan
[*]Corresponding author: Layeq Ali (Email: layeq.ali@is.uol.edu.pk)

**Received: 13/08/2026, Revised: 12/10/2026, Accepted: 14/10/2026**

***Abstract*: Optical ground-to-air detection of small unmanned aerial vehicles (UAVs) supports perimeter security and airspace safety, but a distant UAV may occupy fewer than 16 × 16 pixels, which is 0.0625% of a 640 × 640 detector input. Strided downsampling makes such objects harder to detect, and Intersection over Union (IoU) based box losses are highly sensitive to pixel-level shifts. This paper evaluates YOLOv8-P2-CBAM-NWD, a nano-scale version of YOLOv8 with a stride-4 P2 detection head, a Convolutional Block Attention Module (CBAM) on the shallow P2 branch, and a box loss that combines Complete IoU (CIoU) with the Normalized Gaussian Wasserstein Distance (NWD). Each model was trained only once on 800 single-class drone images and compared against a YOLOv8n baseline. On 200 validation images, the proposed model reached a mean average precision at an IoU of 0.5 (mAP@0.5) of 89.8%, which is below the 92.5% of the baseline. On a sentry test set of 2,500 frames (1,800 drone frames and 700 cloud or clear-sky frames) at a confidence threshold of 0.1, it increased frame-level precision from 96.94% to 98.35% and specificity from 92.14% to 95.86%, and it decreased false alarms from 78.57 to 41.43 per 1,000 negative frames, a 47.3% relative reduction. Frame-level recall went down from 96.78% to 96.11%. Thus, the main advantage is fewer false alarms rather than higher object-level accuracy. Since the threshold was selected on the test set and the frames were not grouped by video, these results are preliminary.**

***Index Terms*: CBAM attention, Counter-UAV, False alarm rate, Normalized Wasserstein distance, Small-object detection, YOLOv8.**

## I. Introduction

OVER the past decade, the development of brushless direct-current (BLDC) propulsion, high-energy-density lithium-polymer (LiPo) batteries, lightweight carbon-fiber composites, and open-source autonomous flight controllers has made unmanned aerial vehicles (UAVs) common in commercial, industrial, and consumer use. Legitimate use cases, including aerial topography, agricultural crop scouting, industrial pipeline inspection, and disaster response, have grown rapidly. At the same time, the spread of micro- and mini-UAVs has created serious asymmetric security threats. Hostile or negligent drone flights have grounded commercial air traffic at major international airports, enabled contraband smuggling over prison perimeters, supported corporate espionage against high-technology manufacturing campuses, and made asymmetric strikes against critical infrastructure possible.

To protect vulnerable airspace, multi-layered counter-UAV (C-UAV) systems have traditionally combined four types of sensors: radio-frequency (RF) scanners, primary radar installations, acoustic sensor arrays, and optical sentry cameras. Optical computer vision provides passive, high-density spatial information without active electromagnetic emissions. It is also the only modality that can visually verify target identity, check whether an airborne object carries a hazardous payload, and track targets in complete radio silence.

Despite these advantages, vision-based drone detection faces a serious physical and algorithmic bottleneck: the small-object resolution dilemma. In realistic perimeter security scenarios, a sentry must detect an inbound micro-drone at a standoff distance of at least 100 to 200 m to leave enough time for automated countermeasures such as directed RF jamming, net capture, or kinetic interception. With a typical optical lens, a drone 30 to 40 cm in diameter can cover fewer than 16 × 16 pixels, and often fewer than 8 × 8 pixels. A 16 × 16 target covers 256 pixels, which is 0.0625% of the 640 × 640 detector input and 0.0123% of a 1920 × 1080 frame.

Under these conditions, conventional deep object detectors such as Faster R-CNN [1], SSD, RetinaNet, and even the standard YOLOv8 [2] lose much of their accuracy. The problem has two root causes, one architectural and one mathematical.

*1) Convolutional stride vanishing:* Standard convolutional backbones reduce the resolution of intermediate feature maps

This work is licensed under a Creative Commons Attribution-ShareAlike 4.0 International License, which permits Unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited

by factors of 8× (P3), 16× (P4), and 32× (P5) to enlarge the receptive field and extract high-level semantic abstractions. For an 8 × 8 pixel target, a stride of 8 reduces the object to about one feature cell, and at strides of 16 and 32 it covers only part of a cell, so its signal is mixed with the surrounding background before it reaches the higher-level detection heads.

*2) IoU sensitivity:* Box regression losses based on Intersection over Union (IoU), such as Complete IoU (CIoU) [3], [4] and Generalized IoU (GIoU) [5], are highly sensitive to small positional errors on tiny targets. For an 8 × 8 box, a 3-pixel shift along one axis lowers the IoU from 1.0 to 40/88 ≈ 0.455, and a 3-pixel shift along both axes lowers it to 25/103 ≈ 0.243. With a shift of 8 pixels or more along one axis, the boxes no longer intersect, so the IoU is exactly 0. In this case, the plain IoU loss has zero gradient, and penalty-based losses such as GIoU and CIoU depend only on their penalty terms, which makes tiny-box regression harder.

To address these challenges, this paper presents YOLOv8-P2-CBAM-NWD, a nano-scale YOLOv8 variant for detecting small drones in sky scenes, and compares it with a YOLOv8n baseline using both object-level and frame-level metrics. The main contributions of this paper are as follows.

- *Theoretical modeling of visual degradation:* We derive the physical, optical, and mathematical chain of feature degradation in ground-to-air vision and show the relation between standoff distance, focal length, pixel pitch, and convolutional stride vanishing.
- *Architectural design (YOLOv8-P2 + CBAM):* We design a high-resolution detection neck with an extra P2 detection head at stride 4 (160 × 160 resolution), combined with Convolutional Block Attention Modules (CBAM) that filter out high-frequency terrestrial clutter.
- *Metric reformulation (NWD loss):* We combine the Normalized Gaussian Wasserstein Distance (NWD) with CIoU in the YOLOv8 box loss (weights 0.7 and 0.3, respectively) and model bounding boxes as two-dimensional (2D) Gaussian distributions, which keeps a smooth similarity measure even when the predicted and true boxes do not overlap.
- *Evaluation at two levels:* We present object-level precision, recall, mAP@0.5, and mAP@0.5:0.95 on a validation set, and frame-level counts, accuracy, precision, recall, specificity, F1-score, and false alarm rate (FAR) on a 2,500-frame test set. In comparison with YOLOv8n, the proposed model reduces the FAR by 47.3% and increases frame-level precision and specificity, while its mAP@0.5 is 2.7 points lower.
- *Transparent protocol and limitations:* We report the data splits, training recipes, and hardware (one Tesla T4 GPU). We also state the weaknesses of the protocol, namely a single run, a frame-level split, and a threshold chosen on the test set, and we list the controls that remain future work.

The rest of the paper is structured as follows. Section II reviews related work. Section III examines the physical, optical, and algorithmic causes of detection failure. Section IV presents the proposed architecture. Section V details the experimental protocol and presents the results, Section VI discusses limitations and future directions, and Section VII concludes the paper.

## II. Related Work

### A. Evolution of Deep Object Detection Architectures

Modern deep object detectors fall into two main groups: two-stage and one-stage frameworks. Two-stage detectors, pioneered by R-CNN, Fast R-CNN, and Faster R-CNN [1], first generate candidate region proposals with a region proposal network (RPN) and then classify and refine the bounding boxes. Faster R-CNN achieves high detection precision, but its multi-stage pipeline is comparatively slow for real-time aerial defense. One-stage detectors, most notably the You Only Look Once (YOLO) family [6], the Single Shot MultiBox Detector (SSD), and RetinaNet, treat object detection as a single end-to-end regression problem and predict class probabilities and bounding-box coordinates simultaneously over dense grids.

The release of YOLOv8 [2] brought a major architectural shift, from anchor-based schemes to an anchor-free, task-aligned dual-head design. YOLOv8 adopts a modified CSPDarknet53 backbone with C2f modules to enrich gradient flow. Like earlier YOLO versions, however, the standard YOLOv8 is tuned for general computer vision benchmarks and places detection heads only at strides 8 (P3), 16 (P4), and 32 (P5), which limits its performance on micro-scale objects. Other real-time detectors include YOLOv6 [7], YOLOv7 [8], and the transformer-based RT-DETR [9]. For vision-based detection and tracking of drones, public benchmarks include Anti-UAV [10] and the Drone-vs-Bird detection challenge [11].

### B. Multi-Scale Feature Pyramids and Attention Mechanisms

To achieve multi-scale object detection, Lin et al. [12] introduced the Feature Pyramid Network (FPN), whose top-down pathway and lateral connections combine high-level semantic features with lower-level spatial features. Liu et al. [13] extended FPN with the Path Aggregation Network (PANet), and Tan et al. [14] later proposed the Bidirectional Feature Pyramid Network (BiFPN). To address severe background interference, Woo et al. [15] proposed the Convolutional Block Attention Module (CBAM), which infers channel and spatial attention maps in sequence. For drone-captured imagery, Zhu et al. [16] added an extra prediction head for tiny objects to YOLOv5 and incorporated CBAM to locate attention regions in dense scenes. Other approaches for small objects include slicing aided hyper inference (SAHI) [17] and cascaded sparse queries on high-resolution features (QueryDet) [18]; neither was evaluated in this study.

### C. Bounding-Box Regression Losses: From IoU to Wasserstein Distance

Classical bounding-box regression relied on the smooth L1 loss or the mean squared error (MSE). Yu et al. [19] proposed the IoU loss, and Zheng et al. proposed the Distance IoU (DIoU) and Complete IoU (CIoU) losses [3] and later analyzed CIoU further [4]. Wang et al. [20], however, showed that IoU-based metrics are highly sensitive to location deviations on tiny objects. To address this limitation, the same authors proposed

the Normalized Gaussian Wasserstein Distance (NWD), which represents bounding boxes as 2D Gaussian distributions. More recent IoU variants, such as Wise-IoU [21], which uses a dynamic focusing mechanism, and Inner-IoU [22], which computes the loss on scaled auxiliary boxes, were not tested in this work. Here, NWD is combined with CIoU in the anchor-free head of YOLOv8.

## III. Physical, Optical, and Algorithmic Failure Analysis

### A. Pinhole Projection and Spatial Sparsity Modeling

Consider an optical camera with a lens of focal length $f$ and an image sensor with pixel pitch $p$. A drone of physical width $w_{\text{real}}$ and height $h_{\text{real}}$, located at a standoff distance $D$ from the optical center, projects onto the camera sensor. Under the standard pinhole projection model, the projected physical dimensions on the sensor plane, $w_{\text{sensor}}$ and $h_{\text{sensor}}$, are

$$w_{\text{sensor}} = \frac{w_{\text{real}} \cdot f}{D},\ h_{\text{sensor}} = \frac{h_{\text{real}} \cdot f}{D} \tag{1}$$

The target dimensions in pixel units, $w_{\text{px}}$ and $h_{\text{px}}$, and the total pixel area $N_{\text{pixels}}$ are

$$w_{\text{px}} = \frac{w_{\text{real}} \cdot f}{D \cdot p},\ h_{\text{px}} = \frac{h_{\text{real}} \cdot f}{D \cdot p} \tag{2}$$

$$N_{\text{pixels}} = w_{\text{px}} \times h_{\text{px}} = \frac{w_{\text{real}} \cdot h_{\text{real}} \cdot f^2}{D^2 \cdot p^2} \tag{3}$$

Equation (3) shows that the target pixel footprint decays quadratically with the standoff distance, that is, as $O(1/D^2)$. For a compact quadcopter (e.g., a DJI Mini 3 with $w_{\text{real}}$ = 0.25 m and $h_{\text{real}}$ = 0.10 m) viewed at $D$ = 120 m through a lens with $f$ = 16 mm and pixel pitch $p$ = 3.45 μm, the projected dimensions are $w_{\text{px}} \approx 9.66$ pixels and $h_{\text{px}} \approx 3.86$ pixels, giving a total target footprint of only about 37 pixels. In a Full HD (1920 × 1080) surveillance frame, which contains 2,073,600 pixels, this micro-target occupies only about 0.0018% of the scene. By comparison, a 16 × 16 target (256 pixels) covers only 0.0625% of the 640 × 640 network input used in this study.

### B. The Convolutional Stride-Vanishing Dilemma

In CNN models such as YOLOv8, the spatial dimensions are progressively reduced by successive convolutional layers with stride 2. Let $S_l$ denote the cumulative downsampling stride at network stage $l$. The effective spatial dimensions of the target at stage $l$ then become

$$w_{\text{feat}}^{(l)} = \frac{w_{\text{px}}}{S_l},\ h_{\text{feat}}^{(l)} = \frac{h_{\text{px}}}{S_l} \tag{4}$$

Table I records the progressive feature erosion of a 10 × 10 pixel drone through the YOLOv8 backbone stages.

TABLE I
Progressive Feature Erosion of a 10 × 10 Pixel Drone Through the YOLOv8 Backbone Stages

| Backbone Stage | Downsampling Stride ($S$) | Feature Map Size (640 × 640 Input) | 10 × 10 Target Size in Feature Map | Feature Information Status |
|---|---|---|---|---|
| C1 (Input stage) | Stride 2 | 320 × 320 | 5.0 × 5.0 pixels | Rich geometric boundaries preserved |
| C2 (Shallow layer) | Stride 4 | 160 × 160 | 2.5 × 2.5 pixels | Core structural features intact (P2 Head Target) |
| C3 (P3 Head) | Stride 8 | 80 × 80 | 1.25 × 1.25 pixels | Marginal boundary fidelity; easily contaminated |
| C4 (P4 Head) | Stride 16 | 40 × 40 | 0.62 × 0.62 sub-pixel | Sub-pixel feature vanishing; high false negatives |
| C5 (P5 Head) | Stride 32 | 20 × 20 | 0.31 × 0.31 sub-pixel | Complete feature extinction; target zeroed out |

As Table I shows, by the time features reach P4 and P5, the micro-drone has collapsed into sub-pixel activations contaminated by the surrounding background pixels. The standard YOLOv8 heads therefore process noise instead of drone signals.

### C. Environmental Background Clutter and Natural Distractors

In ground-to-air tracking, drones appear against heavy visual clutter, including tree canopies with swaying foliage, architectural edges, and moving clouds. Birds can appear at similar scales of less than 16 pixels and move very rapidly, which makes them a frequent source of false alarms. The negative images in this study, however, contain only clouds and clear sky, so bird rejection is not evaluated here.

## IV. Proposed Method: The YOLOv8-P2-CBAM-NWD Architecture

To address stride vanishing, background clutter interference, and IoU regression instability, we design the YOLOv8-P2-CBAM-NWD framework. This section explains the network topology, the P2 neck routing, the CBAM attention mechanism, and the NWD formulation.

### A. High-Resolution P2 Detection Head Topology

The standard YOLOv8 consists of a backbone, an FPN-PAN neck, and three decoupled detection heads at P3, P4, and P5. The proposed architecture adds a fourth high-resolution detection stream anchored at the C2 backbone stage, as follows.

#### 1) Backbone Extraction

The input image $I \in \mathbb{R}^{640 \times 640 \times 3}$ passes through convolutional and C2f blocks, which at the nano scale (depth multiple 0.33, width multiple 0.25) produce the feature hierarchy C2 (stride 4, 160 × 160 × 32), C3 (stride 8, 80 × 80 × 64), C4 (stride 16, 40 × 40 × 128), and C5 (stride 32, 20 × 20 × 256).

#### 2) Top-Down High-Resolution Pathway

In the FPN neck, the P3 feature map (80 × 80) is upsampled by a factor of 2 to 160 × 160 and then concatenated with the

shallow C2 feature map from the backbone, which is first refined by CBAM in the proposed model. The fused tensor passes through one more C2f block, which merges fine spatial detail with contextual semantics to produce the feature map P2. A bottom-up pathway then downsamples P2 with a stride-2 convolution and fuses it with P3, P4, and P5 in turn.

*3) Decoupled Four-Head Detection*

The neck generates four multiscale tensors: P2 (stride 4), P3 (stride 8), P4 (stride 16), and P5 (stride 32).

### B. Dual-Domain Attention Routing via CBAM

To suppress high-frequency background foliage and cloud gradients, a single CBAM module is added to the lateral connection from backbone stage C2 (32 channels at the nano scale) to the P2 fusion node. CBAM sequentially infers a one-dimensional channel attention map $M_c \in \mathbb{R}^{C \times 1 \times 1}$ and a two-dimensional spatial attention map $M_s \in \mathbb{R}^{1 \times H \times W}$:

$$M_c(F) = \sigma(\mathrm{MLP}(\mathrm{AvgPool}(F)) + \mathrm{MLP}(\mathrm{MaxPool}(F))) \quad (5)$$

$$M_s(F') = \sigma\left(f^{(7\times7)}([\mathrm{AvgPool}(F'); \mathrm{MaxPool}(F')])\right) \quad (6)$$

where $F$ is the input feature map, $F'$ is the channel-refined feature map, σ is the sigmoid function, MLP is the shared multilayer perceptron, AvgPool and MaxPool denote average pooling and max pooling, respectively, and $f^{(7\times7)}$ is a convolution with a 7 × 7 kernel. The refined feature map is $F'' = M_s(F') \otimes F'$, where ⊗ denotes element-wise multiplication. In our implementation, the shared MLP has a hidden width of 8, and the spatial branch uses a 7 × 7 convolution. The module is intended to enhance compact, high-contrast structures such as drone bodies and to weaken homogeneous cloud and foliage regions; this effect was not measured directly in this study.

### C. Normalized Gaussian Wasserstein Distance Formulation

For micro-drone detection, the standard IoU behaves in a discrete, discontinuous way. Let $B_p = (cx_p, cy_p, w_p, h_p)$ and $B_g = (cx_g, cy_g, w_g, h_g)$ represent the predicted and ground-truth bounding boxes, respectively, where $(cx, cy)$ is the box center and $w$ and $h$ are the box width and height. To remove the sensitivity threshold, each bounding box is modeled as a 2D Gaussian distribution $N(\mu, \Sigma)$ inscribed within the box boundaries, with probability density function

$$f(x \mid \mu, \Sigma) = \frac{1}{2\pi|\Sigma|^{1/2}} \cdot \exp\left(-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)\right) \quad (7)$$

where the mean vector μ and the covariance matrix Σ are

$$\mu = [cx, cy]^T,\ \Sigma = \begin{bmatrix} \frac{w^2}{4} & 0 \\ 0 & \frac{h^2}{4} \end{bmatrix} \quad (8)$$

The optimal transport distance between the predicted distribution $N_p(\mu_p, \Sigma_p)$ and the ground-truth distribution $N_g(\mu_g, \Sigma_g)$ is computed with the second-order Wasserstein distance $W_2^2(N_p, N_g)$:

$$W_2^2(N_p, N_g) = \|\mu_p - \mu_g\|_2^2 + \mathrm{Tr}\left(\Sigma_p + \Sigma_g - 2\left(\Sigma_p^{1/2}\Sigma_g\Sigma_p^{1/2}\right)^{1/2}\right) \quad (9)$$

where Tr(·) is the matrix trace. Because the covariance matrices $\Sigma_p$ and $\Sigma_g$ are diagonal, (9) simplifies to an exact closed-form expression:

$$W_2^2(N_p, N_g) = (cx_p - cx_g)^2 + (cy_p - cy_g)^2 + (w_p/2 - w_g/2)^2 + (h_p/2 - h_g/2)^2 \quad (10)$$

After exponential normalization, the Normalized Wasserstein Distance is obtained:

$$\mathrm{NWD}(N_p, N_g) = \exp\left(-\frac{\sqrt{W_2^2(N_p, N_g)}}{C}\right) \quad (11)$$

where $C$ is a normalizing constant in pixels. Wang et al. [20] set $C$ to the average absolute object size of their dataset. In this study, $C = 50.0$ for the 640 × 640 input. This value was chosen empirically to keep the similarity from saturating for box offsets of tens of pixels, and no sweep over $C$ was performed; a sensitivity analysis is deferred to future work. The corresponding NWD regression loss $L_{\mathrm{NWD}}$ is then given by

$$L_{\mathrm{NWD}} = 1 - \mathrm{NWD}(N_p, N_g) \quad (12)$$

The combined training loss $L_{\mathrm{total}}$ is

$$L_{\mathrm{total}} = \alpha_{\mathrm{cls}} \cdot L_{\mathrm{cls}} + \alpha_{\mathrm{box}} \cdot L_{\mathrm{box}} + \alpha_{\mathrm{dfl}} \cdot L_{\mathrm{dfl}} \quad (13)$$

where $L_{\mathrm{cls}}$, $L_{\mathrm{box}}$, and $L_{\mathrm{dfl}}$ denote the classification, bounding-box, and distribution focal losses, and $\alpha_{\mathrm{cls}}$, $\alpha_{\mathrm{box}}$, and $\alpha_{\mathrm{dfl}}$ are their weights, which were kept at the Ultralytics default values. The bounding-box loss combines the NWD and CIoU terms as follows:

$$L_{\mathrm{box}} = \lambda \cdot L_{\mathrm{NWD}} + (1 - \lambda) \cdot L_{\mathrm{CIoU}} \quad (14)$$

with $\lambda = 0.7$, that is, $L_{\mathrm{box}} = 0.7L_{\mathrm{NWD}} + 0.3L_{\mathrm{CIoU}}$. In the implementation, this composite term replaces the IoU function of Ultralytics, so it is used by the task-aligned label assigner as well, and not only by the box loss. The baseline M0 and the variants M1 and M3 use the default CIoU loss.

## V. Experiments and Results

This section describes the dataset, training procedure, and metrics, and then reports object-level and frame-level results for the YOLOv8n baseline (M0) and the P2-based variants M1 to M4. Each model was trained only once, so all numbers come from a single training run.

### A. Datasets and Data Splits

All experiments use a single-class (drone) image dataset in YOLO format, divided into training and validation folders, together with a separate set of drone-free sky images showing cumulus clouds, cirrus clouds, and clear sky. The training set consists of the first 800 images of the training folder. The validation set consists of the first 200 images of the validation folder, which contain 219 annotated drones and no background-only images. The sentry test set consists of 2,500 frames: the first 1,800 images of the same validation folder, all of which contain a drone, and 700 drone-free cloud or clear-sky images used as negatives. Files were taken in directory-listing order, and the data were split by frame, not by video sequence. Since the validation images and the positive test frames come from

the same folder, the 200 validation images are very likely included among the 1,800 positive test frames, and the validation set was also used to select the best checkpoint. The negative images contain no birds, aircraft, or buildings. The drone image dataset is publicly available at https://drive.google.com/file/d/1iwVljGePZrKFq6jdqVgGu7G-vkqWq3fm/view?usp=sharing.

### B. Training and Evaluation Protocol

Table II summarizes the training settings of each model. All models use the nano scale of YOLOv8 (depth multiple 0.33, width multiple 0.25), an input size of 640 × 640, a batch size of 16, and random seed 42. The baseline M0 is the COCO-pretrained YOLOv8n, fine-tuned with stochastic gradient descent (SGD) for 20 epochs. The P2 variants M1 to M4 were initialized by copying COCO-pretrained YOLOv8n weights into all layers with matching names and shapes, including the P3 to P5 head branches, while the P2 head and CBAM were trained from scratch with AdamW. Both recipes used a cosine learning-rate schedule with a final factor of 0.01, mosaic augmentation with probability 1.0 that was disabled for the last 5 epochs, and the other default Ultralytics augmentations. The notebook configures M1 to M4 for 25 epochs, but the saved training log of M4 contains only 20 epochs, so the exact epoch count of M1 to M3 cannot be confirmed. The baseline and the P2 variants therefore differ in optimizer, learning rate, warm-up, and initialization, and the comparison is not a controlled same-recipe study. Training and evaluation were performed on one NVIDIA Tesla T4 GPU (Google Colab) with PyTorch 2.11.0 and Ultralytics 8.4.175 [2]. For each model, the checkpoint with the best validation fitness was kept.

TABLE II
TRAINING CONFIGURATION OF THE BASELINE AND THE P2 VARIANTS

| Setting | M0 (baseline) | M1 to M4 (P2 variants) |
|---|---|---|
| Model scale | YOLOv8n (0.33/0.25) | Nano P2 (0.33/0.25) |
| Initialization | COCO-pretrained YOLOv8n | COCO weights in matching layers; P2 head and CBAM from scratch |
| Optimizer | SGD (momentum 0.937) | AdamW |
| Initial learning rate | 0.01 | 0.003 |
| Weight decay | 0.0005 | 0.01 |
| Schedule | Cosine, final factor 0.01 | Cosine, final factor 0.01 |
| Warm-up epochs | 2 | 3 |
| Epochs | 20 | 25 configured (20 in M4 log) |
| Batch size / input | 16 / 640 × 640 | 16 / 640 × 640 |
| Augmentation | Mosaic 1.0, off for last 5 epochs; other defaults | Same as M0 |
| Box loss | CIoU | CIoU (M1, M3); 0.3 CIoU + 0.7 NWD (M2, M4) |
| Seed / runs | 42 / 1 | 42 / 1 |
| Hardware / software | Tesla T4; Ultralytics 8.4.175 | Same as M0 |

Object-level metrics were computed with the Ultralytics validator on the 200 validation images, and frame-level metrics were computed on the 2,500 test frames. Each frame was processed with a minimum confidence of 0.05, and a frame was marked as positive when its highest detection confidence reached the decision threshold $\tau$. The threshold $\tau = 0.1$ was selected as the value that maximized the F1-score of M4 in a sweep from 0.10 to 0.80 on the test set itself, and the same $\tau$ was applied to M0. Because $\tau$ was tuned on the test data and 0.10 is the lower end of the sweep, the frame-level results are optimistically biased for M4. No confidence intervals were computed.

### C. Metric Definitions

We report two types of metrics that answer different questions and should not be compared directly.

*1) Frame-level metrics (test set):* A positive frame contains at least one annotated drone, and a negative frame contains none. A true positive (TP) is a positive frame whose maximum detection confidence is greater than or equal to $\tau$, whereas a false negative (FN) is a positive frame whose maximum detection confidence is below $\tau$. A false positive (FP) is a negative frame with at least one detection at or above $\tau$, and a true negative (TN) is a negative frame whose maximum detection confidence is below $\tau$. A frame-level TP does not check whether the detection lies on the drone. From these counts, Precision = TP/(TP + FP), Recall = TP/(TP + FN), Accuracy = (TP + TN)/(TP + TN + FP + FN), Specificity = TN/(TN + FP), F1 = 2 · Precision · Recall/(Precision + Recall), and FAR = 1000 · FP/(TN + FP), the number of false alarms per 1,000 negative frames.

*2) Object-level metrics (validation set):* Precision, recall, mAP@0.5, and mAP@0.5:0.95 follow the Ultralytics definitions, in which a predicted box is a true positive when it matches an annotated box at the given IoU threshold. mAP@0.5 is the area under the precision-recall curve at an IoU of 0.5, and mAP@0.5:0.95 is the average of this area over IoU thresholds from 0.5 to 0.95 in steps of 0.05. Object-level precision and recall are reported at the operating point selected by the Ultralytics validator.

### D. Object-Level Validation and Ablation

Table III shows the object-level results. The baseline M0 has the highest recall (88.7%), mAP@0.5 (92.5%), and mAP@0.5:0.95 (57.0%). Adding the P2 head (M1) improves precision to 90.5% but reduces recall to 80.8% and mAP@0.5 to 90.6%. Adding NWD to the P2 model (M2) gives the highest precision (91.3%), with a recall of 79.5%. The full model M4 achieves a precision of 90.3%, a recall of 80.9%, an mAP@0.5 of 89.8%, and an mAP@0.5:0.95 of 49.4%, which is 2.7 and 7.6 points below M0 on the two mAP measures. The evaluation of M3 (P2 and CBAM without NWD) failed with a software error ('Linear' object has no attribute 'in_channels'), so no metrics are available for M3 and the separate effect of CBAM cannot be isolated. The differences among M1, M2, and M4 are about 1 point or less, which a single run cannot separate from training noise. Since M0 also used a different optimizer and learning rate, part of the gap between M0 and the P2 variants may come from the training recipe rather than from the architecture.

TABLE III
OBJECT-LEVEL ABLATION ON THE VALIDATION SET (200 IMAGES, 219 DRONES)

| Model | P2 | CBAM | NWD | P (%) | R (%) | mAP@ 0.5 (%) | mAP@ 0.5:0.95 (%) |
|---|---|---|---|---|---|---|---|
| M0 (YOLOv8n) | No | No | No | 89.0 | 88.7 | 92.5 | 57.0 |
| M1 | Yes | No | No | 90.5 | 80.8 | 90.6 | 50.0 |
| M2 | Yes | No | Yes | 91.3 | 79.5 | 90.1 | 50.3 |
| M3 | Yes | Yes | No | n/a | n/a | n/a | n/a |
| M4 (proposed) | Yes | Yes | Yes | 90.3 | 80.9 | 89.8 | 49.4 |

n/a: the validation run of M3 failed with a software error, so no metrics are available.

Model size, computation, and speed are reported in Table IV. Adding the P2 head increases the computation from 8.6 to 12.2 GFLOPs, while the parameter count decreases slightly from 3.13 M to 2.92 M, mainly because the Ultralytics detection head sets the width of its classification branches from the number of channels in its first (P2) input. On the Tesla T4, the inference time per image increases from 5.4 ms (M0) to 8.2 ms (M4). Preprocessing and postprocessing times, the latter including non-maximum suppression (NMS), vary between runs from 1.3 to 3.8 ms, so the totals of the three stages (11.6 ms for M0 and 12.5 ms for M4) should be considered indicative only. All times were measured in PyTorch during validation with a batch size of 16, and no embedded or TensorRT measurements were made.

TABLE IV
MODEL COMPLEXITY AND PER-IMAGE TIME ON A TESLA T4 GPU (PYTORCH, BATCH 16)

| Model | Params | GFLOPs | Pre (ms) | Infer. (ms) | NMS (ms) | Sum (ms) |
|---|---|---|---|---|---|---|
| M0 | 3,132,707 | 8.6 | 3.6 | 5.4 | 2.6 | 11.6 |
| M1 | 2,921,172 | 12.2 | 3.8 | 7.9 | 2.1 | 13.8 |
| M2 | 2,921,172 | 12.2 | 2.4 | 7.8 | 1.3 | 11.5 |
| M3 | 2,921,782 | n/a | n/a | n/a | n/a | n/a |
| M4 | 2,921,782 | 12.2 | 2.7 | 8.2 | 1.6 | 12.5 |

Params: parameters. Pre: preprocessing. Infer.: network forward pass. NMS: postprocessing including non-maximum suppression. Sum: total of the three stages. Parameters are for the fused models.

Fig. 1 shows the validation mAP@0.5 of M0 and M4 after each epoch, taken from the Ultralytics training logs. Both curves fluctuate during the first ten epochs, and M4 approaches M0 towards the end of training without exceeding it. The training losses of the two models are not compared, because M4 uses a different box loss (0.3 CIoU + 0.7 NWD), so its loss values are not on the same scale as those of M0.

### *E. Frame-Level Sentry Evaluation*

Table V and Fig. 2 report the frame-level results at $\tau = 0.1$. M4 raised alarms on 29 of the 700 negative frames, compared with 55 for M0, so the FAR decreased from 78.57 to 41.43 per 1,000 negative frames, a relative reduction of 47.3% ((78.57 − 41.43)/78.57). Specificity increased from 92.14% to 95.86%, and frame-level precision improved from 96.94% to 98.35%. M4 missed 70 of the 1,800 positive frames, against 58 for M0, so frame-level recall fell from 96.78% to 96.11%.

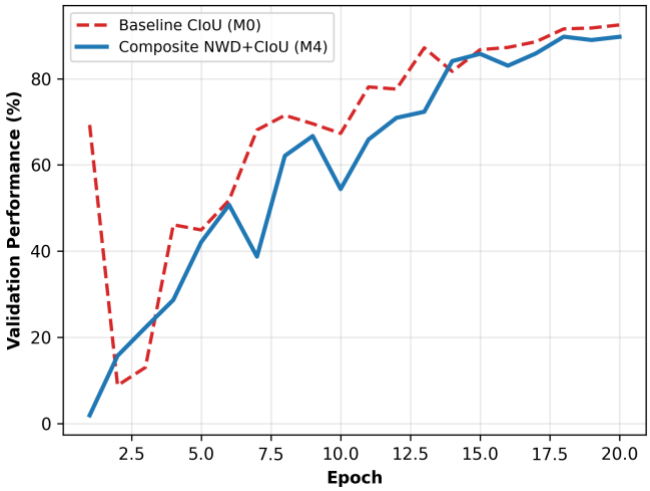



Fig. 1. Validation mAP@0.5 (%) after each epoch for M0 (dashed) and M4 (solid), from the Ultralytics training logs on the 200-image validation set. The y-axis label "Validation Performance" denotes mAP@0.5; the original panel title, which called this quantity mean IoU, was incorrect and has been removed.

M4 increased accuracy from 95.48% to 96.04% and the F1-score from 96.86% to 97.22%, so both changed by less than 1 point. However, because the negative frames include only clouds and clear sky, these results indicate fewer false alarms on sky backgrounds; they do not demonstrate robustness to birds or other airborne distractors.

TABLE V
FRAME-LEVEL RESULTS ON THE 2,500-FRAME TEST SET AT $\tau = 0.1$

| Metric | M0 | M4 | M4 − M0 |
|---|---|---|---|
| TP (of 1,800 positive frames) | 1,742 | 1,730 | −12 |
| FN | 58 | 70 | +12 |
| FP (of 700 negative frames) | 55 | 29 | −26 |
| TN | 645 | 671 | +26 |
| Accuracy (%) | 95.48 | 96.04 | +0.56 |
| Precision (%) | 96.94 | 98.35 | +1.41 |
| Recall (%) | 96.78 | 96.11 | −0.67 |
| Specificity (%) | 92.14 | 95.86 | +3.72 |
| F1-score (%) | 96.86 | 97.22 | +0.36 |
| FAR (per 1,000 negative frames) | 78.57 | 41.43 | −37.14 |

Differences are in counts or percentage points. The relative FAR reduction is 37.14/78.57 = 47.3%.

Fig. 3 compares the five frame-level metrics of the two models. Fig. 4 shows the frame-level F1-score as a function of the threshold $\tau$. The two curves are close for $\tau$ up to about 0.3, M0 is slightly higher over most of the range from about 0.15 to 0.6, and both fall sharply above $\tau \approx 0.5$. The best F1-scores reported in the legend are 97.2% for M4 and 97.3% for M0; however, they were taken over different ranges of $\tau$, as noted in the caption. Fig. 5 shows the frame-level precision-recall curves obtained by sweeping $\tau$. Both models sustain frame-level precision close to 100% up to about 80% recall, and at the highest recall levels M4 maintains slightly higher precision than M0, which is consistent with its lower FAR.

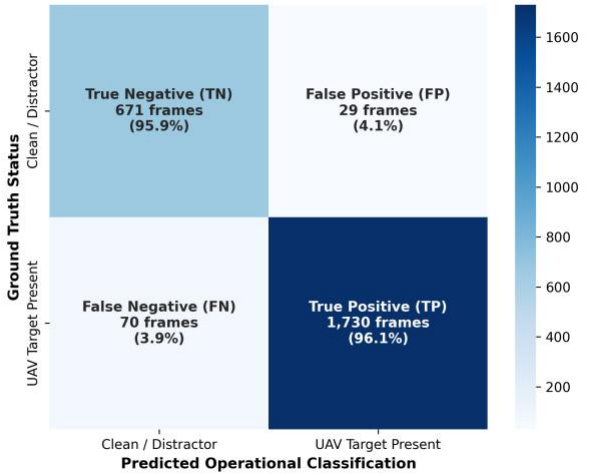



Fig. 2. Frame-level confusion matrix of M4 on the 2,500 test frames at τ = 0.1. Row percentages are shown in parentheses.

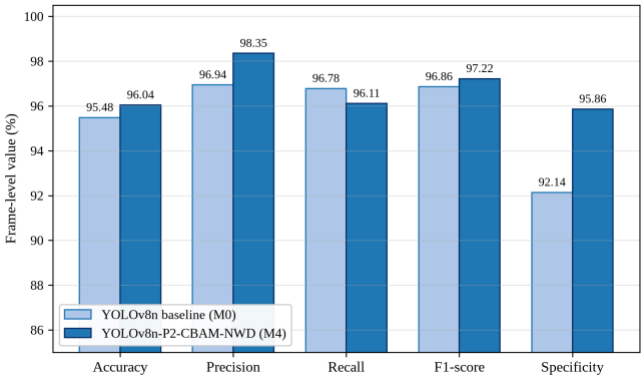



Fig. 3. Frame-level accuracy, precision, recall, F1-score, and specificity of M0 and M4 at τ = 0.1 (2,500 test frames), plotted from the values in Table V. The vertical axis begins at 85%.

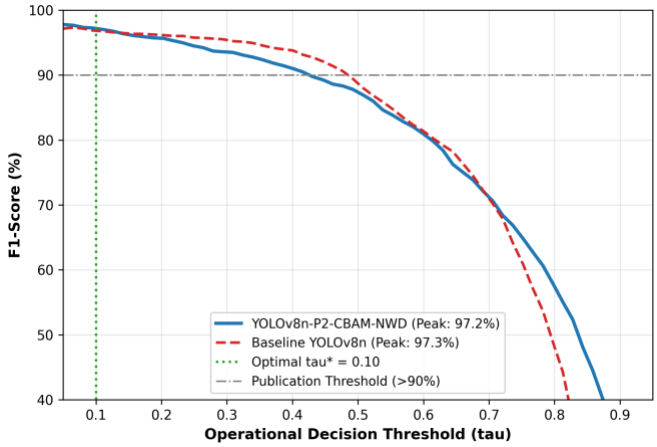



Fig. 4. Frame-level F1-score versus decision threshold τ for M0 and M4. The legend peak for M4 (97.2%) is the maximum F1-score over thresholds from 0.10 to 0.80, the sweep used to select τ = 0.1, whereas the legend peak for M0 (97.3%) is the maximum over thresholds from 0.05 to 0.95. The dash-dot line at 90% is only a plotting reference.

### F. Frame-Level Recall by Target Size

To study small targets, each annotated drone in the 1,800 positive test frames was assigned to a size bin using $s = \sqrt{(w \cdot h)}$, where $w$ and $h$ are the box width and height in pixels at the 640 × 640 input scale: micro ($s < 16$ pixels), small ($16 \le s \le 32$ pixels), and medium ($s > 32$ pixels). These bins were defined for this study and differ from the COCO small, medium, and large area ranges. The measure is a frame-level recall per annotated drone: a drone is counted as detected when the maximum detection confidence in its frame reaches τ, without checking box overlap. It is therefore not an average precision, and the label $mAP_{small}$ is no longer used. Fig. 6 presents the results. For micro targets, M4 detected 66.2% of the drones, compared with 70.3% for M0. Both models reached 96.8% on small targets, while on medium targets M4 detected 97.8% of the drones compared with 98.3% for M0. The proposed model therefore did not improve recall on micro targets in this test. The number of drones in each bin was not recorded.

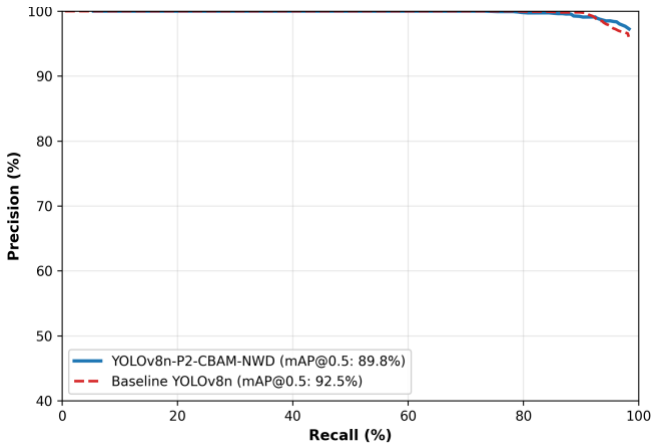



Fig. 5. Frame-level precision-recall curves of M0 and M4 on the 2,500 test frames, obtained by sweeping the threshold τ from 0.05 to 0.95. The mAP@0.5 values in the legend are the object-level validation values from Table III, not the areas under these curves.

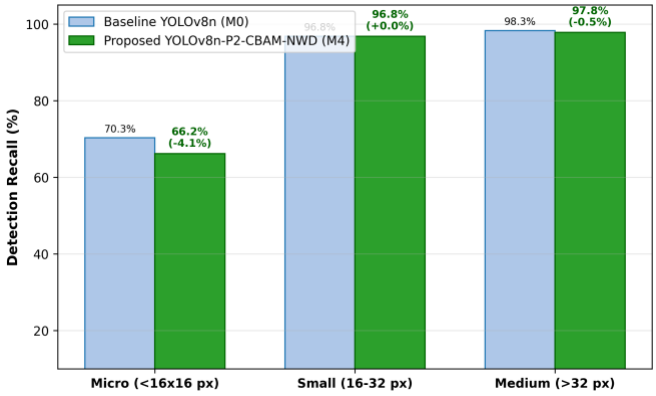



Fig. 6. Frame-level recall per annotated drone by target size, with $s = \sqrt{(w \cdot h)}$ at 640 × 640, for M0 and M4 at τ = 0.1. Micro: $s < 16$ px; small: $16 \le s \le 32$ px; medium: $s > 32$ px. Numbers in parentheses are differences in percentage points.

## VI. Limitations and Future Directions

This section first states the limitations of the current study and then describes possible directions for future research.

### A. Limitations of This Study

The conclusions of this paper are subject to several limitations. First, the comparison covers only the nano scale. Stronger baselines, such as a YOLOv8s-P2 model, slicing aided hyper inference (SAHI) [17], and QueryDet [18], were not tested, and neither were recent IoU variants such as Wise-IoU [21] and Inner-IoU [22]. Second, the baseline and the P2

variants were trained with different optimizers, learning rates, and initialization, and the number of epochs used for training M1 to M3 could not be confirmed from the logs. Third, each model was trained once with one seed, and no confidence intervals were estimated, so differences of about 1 point should not be interpreted as meaningful. Fourth, the data were split by individual frame without grouping video sequences, the validation images very likely overlap with the positive test frames, and the decision threshold was tuned on the test set. Each of these factors may make the reported numbers optimistic.

Fifth, the negative test frames include only clouds and clear sky, so the false-alarm results do not cover birds, aircraft, or urban clutter. Sixth, the evaluation of the CBAM-only variant (M3) failed, so the separate contribution of CBAM could not be assessed, and the NWD constant $C = 50.0$ was not tuned. Seventh, the paper shows no real detection examples or failure cases, and all timings were measured on a Tesla T4 GPU rather than on embedded hardware. A follow-up study needs a video-level split with a held-out test set, a threshold chosen on validation data only, identical training recipes, multiple seeds with confidence intervals, and public benchmarks such as Anti-UAV [10] and Drone-vs-Bird [11], which include bird distractors.

### *B. Future Directions*

#### *1) Neuromorphic Event-Based Sensing*

Conventional CMOS cameras have fixed exposure windows (30 to 60 FPS) and suffer severe motion blur during rapid drone maneuvers. Bio-inspired event cameras record asynchronous per-pixel illumination changes at microsecond resolution (more than 10,000 FPS equivalent) with dynamic ranges exceeding 120 dB [23]. Fusing event streams with shallow P2 feature maps is a promising direction.

#### *2) Spatio-Temporal Consistency and Kinematic Tracklet Association*

Combining YOLOv8-P2-CBAM-NWD with lightweight temporal motion association, such as Kalman-guided ByteTrack [24], could let sentries use kinematic trajectories to suppress residual false alarms.

#### *3) Sim-to-Real Domain Adaptation*

Combining photorealistic physics engines (Unreal Engine 5 and AirSim) with domain-adversarial neural architectures is a possible way to enlarge the training data with synthetic scenes.

#### *4) Embedded Edge Deployment*

The models were not deployed on embedded hardware in this study. Exporting the network to an embedded accelerator, for example with 8-bit integer quantization and quantization-aware training [25], and measuring latency, power, and accuracy on the device is left as future work.

## VII. Conclusion

Detecting small, low-altitude UAVs in cluttered visual airspace is essential for public safety and security. This paper analyzed two known difficulties of standard detectors on tiny targets, convolutional stride vanishing and IoU sensitivity. It then evaluated YOLOv8-P2-CBAM-NWD, a nano-scale YOLOv8 variant with a stride-4 P2 head, a CBAM module on the shallow P2 branch, and a box loss that combines NWD and CIoU.

In a single-run study on a 2,500-frame sentry test set, the proposed model reduced the false alarm rate from 78.57 to 41.43 per 1,000 negative frames, a 47.3% relative decrease, and increased frame-level precision from 96.94% to 98.35% and specificity from 92.14% to 95.86%, at the expense of a small drop in frame-level recall from 96.78% to 96.11%. At the object level, the proposed model did not outperform the YOLOv8n baseline: its mAP@0.5 was 89.8% versus 92.5%, and its frame-level recall on micro targets was 66.2% versus 70.3%. The proposed design therefore appears useful mainly for reducing false alarms on sky backgrounds. Because of the frame-level split, the threshold tuned on the test set, and the unequal training recipes, these findings should be verified with video-level splits, matched training, multiple seeds, and stronger baselines.

## CONFLICTS OF INTEREST

The authors declare no conflicts of interest to report regarding the present study.

## AUTHOR CONTRIBUTIONS

Conceptualization, L.A., I.A.K., and S.Y.; methodology, S.Y. and I.A.K.; software, L.A., and S.Y.,.; validation, I.A.K.; formal analysis, S.Y., I.A.K., and L.A.; investigation, L.A. and I.A.K.; resources, I.A.K.; data curation, S.Y.; writing—original draft preparation, L.A., and I.A.K.; writing—review and editing, I.A.K.; supervision, S.Y.; project administration, S.Y. and I.A.K.

## FUNDING STATEMENT

This research received no external funding.

## INSTITUTIONAL REVIEW BOARD STATEMENT

Not applicable.

## INFORMED CONSENT STATEMENT

Not applicable.

## DATA AVAILABILITY STATEMENT

Data is available on reasonable request.

## CONSENT FOR PUBLICATION

All authors have read and approved the final version of the manuscript and consent to its publication.

## DECLARATION OF GENERATIVE AI IN SCIENTIFIC WRITING

For preparing this manuscript, the authors have only made use of generative artificial intelligence systems to enhance language proficiency, grammar, and readability.

## References

[1] S. Ren *et al.*, "Faster R-CNN: Towards real-time object detection with region proposal networks," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, no. 6, pp. 1137-1149, Jun. 2017, doi: 10.1109/TPAMI.2016.2577031.

[2] G. Jocher *et al.*, "Ultralytics YOLOv8," version 8.0.0, software, 2023. [Online]. Available: https://github.com/ultralytics/ultralytics

[3] Z. Zheng *et al.*, "Distance-IoU loss: Faster and better learning for bounding box regression," in *Proc. AAAI Conf. Artif. Intell. (AAAI)*, vol. 34, no. 7, 2020, pp. 12993-13000, doi: 10.1609/aaai.v34i07.6999.

[4] Z. Zheng *et al.*, "Enhancing geometric factors in model learning and inference for object detection and instance segmentation," *IEEE Trans. Cybern.*, vol. 52, no. 8, pp. 8574-8586, Aug. 2022, doi: 10.1109/TCYB.2021.3095305.

[5] H. Rezatofighi *et al.*, "Generalized intersection over union: A metric and a loss for bounding box regression," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2019, pp. 658-666.

[6] J. Redmon *et al.*, "You only look once: Unified, real-time object detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2016, pp. 779-788, doi: 10.1109/CVPR.2016.91.

[7] C. Li *et al.*, "YOLOv6: A single-stage object detection framework for industrial applications," arXiv preprint arXiv:2209.02976, 2022.

[8] C.-Y. Wang *et al.*, "YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2023, pp. 7464-7475.

[9] Y. Zhao *et al.*, "DETRs beat YOLOs on real-time object detection," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2024, pp. 16965-16974.

[10] N. Jiang *et al.*, "Anti-UAV: A large-scale benchmark for vision-based UAV tracking," *IEEE Trans. Multimedia*, vol. 25, pp. 486-500, 2023.

[11] A. Coluccia *et al.*, "Drone-vs-Bird detection challenge at IEEE AVSS2021," in *Proc. 17th IEEE Int. Conf. Adv. Video Signal Based Surveill. (AVSS)*, 2021, pp. 1-8, doi: 10.1109/AVSS52988.2021.9663844.

[12] T.-Y. Lin *et al.*, "Feature pyramid networks for object detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2017, pp. 2117-2125.

[13] S. Liu *et al.*, "Path aggregation network for instance segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2018, pp. 8759-8768.

[14] M. Tan *et al.*, "EfficientDet: Scalable and efficient object detection," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2020, pp. 10781-10790.

[15] S.R. Babary, and R. Khalid. "Leveraging Transformer-Based Natural Language Processing for Adaptive Language Learning in Digital Humanities Environments." *Journal of Artificial Intelligence and Computing* 4, no. 1, p. 36-43, 2026.

[16] X. Zhu *et al.*, "TPH-YOLOv5: Improved YOLOv5 based on transformer prediction head for object detection on drone-captured scenarios," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. Workshops (ICCVW)*, 2021, pp. 2778-2788, doi: 10.1109/ICCVW54120.2021.00312.

[17] N. Fawad, M. N. Khan, A. Addas, Q. Awais, and N. Ayub. "Game mechanics and artificial intelligence personalization: A framework for adaptive learning systems." *Education Sciences* 15, no. 3, p. 301, 2025.

[18] C. Yang *et al.*, "QueryDet: Cascaded sparse query for accelerating high-resolution small object detection," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2022, pp. 13668-13677.

[19] J. Yu *et al.*, "UnitBox: An advanced object detection network," in *Proc. 24th ACM Int. Conf. Multimedia (MM)*, 2016, pp. 516-520, doi: 10.1145/2964284.2967274.

[20] J. Wang *et al.*, "A normalized Gaussian Wasserstein distance for tiny object detection," arXiv preprint arXiv:2110.13389, 2021.

[21] Z. Tong *et al.*, "Wise-IoU: Bounding box regression loss with dynamic focusing mechanism," arXiv preprint arXiv:2301.10051, 2023.

[22] H. Zhang *et al.*, "Inner-IoU: More effective intersection over union loss with auxiliary bounding box," arXiv preprint arXiv:2311.02877, 2023.

[23] G. Gallego *et al.*, "Event-based vision: A survey," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 44, no. 1, pp. 154-180, Jan. 2022, doi: 10.1109/TPAMI.2020.3008413.

[24] Y. Zhang *et al.*, "ByteTrack: Multi-object tracking by associating every detection box," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, 2022, pp. 1-21, doi: 10.1007/978-3-031-20047-2_1.

[25] B. Jacob *et al.*, "Quantization and training of neural networks for efficient integer-arithmetic-only inference," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2018, pp. 2704-2713, doi: 10.1109/CVPR.2018.00286.