

## APPLIED RESEARCH

# Fast Drone Detection With Optimized Feature Capture and Modeling Algorithms

XIAOHAN TU<sup>1</sup>, (Member, IEEE), CHUANHAO ZHANG<sup>1</sup>, HAIYAN ZHUANG<sup>1</sup>,  
SIPING LIU<sup>2</sup>, AND RENFA LI<sup>3</sup>, (Senior Member, IEEE)

<sup>1</sup>Department of Image and Network Investigation, Zhengzhou Police University, Zhengzhou 450053, China

<sup>2</sup>School of Cyber Science and Engineering, Zhengzhou University, Zhengzhou 450002, China

<sup>3</sup>College of Computer Science and Electronic Engineering, Hunan University, Changsha 410082, China

Corresponding author: Siping Liu (liusiping@hnu.edu.cn)


This work was supported in part by Henan Natural Science Foundation under Grant 242300420693, in part by the Key Scientific Research Project Plan of Colleges and Universities in Henan Province under Grant 23A520042 and Grant 23B520019, in part by the Science and Technology Plan of PRC Ministry of Public Security under Grant 2023JSYJC28, in part by the Fundamental Research Funds for Central Universities of Zhengzhou Police University under Grant 2022TJJBKY002 and Grant 2023TJJBKY012, in part by the Key Research and Development and Promotion Special Project of Henan Province (Scientific and Technological Research) under Grant 232102240015, and in part by the National Natural Science Foundation of China under Grant 61932010.

**ABSTRACT** Detecting drones is a complex challenge, primarily due to their small feature size for extraction and variable lighting conditions. It is crucial to effectively capture and model features for drone detection. To accurately detect drones, we propose feature capture and modeling modules. The feature capture module has a minimal number of FLOPs and parameters. It consists of both local and global attention branches, which capture contextual information and global dependencies across the entire feature set. Complementing this, our feature modeling module innovatively calculates attention maps without any additional parameters. This module augments the capability of the feature capture mechanism to represent complex patterns more effectively. Finally, to ensure rapid deployment, we convert the proposed models to machine codes by introducing a compiler, accelerating drone detection. The compiler unifies inter- and intra-operator scheduling with task abstraction. It optimizes the codes for hardware. In compiling time, the effective schedule is performed. The compilation ensures that drone detection is real-time and accuracy remains unchanged. Through rigorous testing, our methods have demonstrated superiority over most current ones in several metrics, including accuracy, parameter quantities, FLOPs, average FPS, visual effects, and latency. Our method yields at least 23.5% and 12.47% higher  $AR_M$  than existing methods on DUT-Anti-UAV and Online Drone datasets. Our inference speed is at least 6.49% higher than other methods on NVIDIA RTX 2080 Ti GPU.

**INDEX TERMS** Deep neural networks (DNNs), drone detection, feature capture module (FCM), feature modeling module (FMM), optimization.

## I. INTRODUCTION

Drones, also known as unmanned aerial vehicles (UAVs), have become increasingly common and have brought about many advances in industries like photography, agriculture, and surveillance [1]. Drone usage is becoming more commonplace, but it also brings up concerns about misuse, security, and privacy. For instance, drones might target crucial

The associate editor coordinating the review of this manuscript and approving it for publication was Genoveffa Tortora .

infrastructure such as power stations, airports, or government facilities. Drones hovering above crowded public events or gatherings may raise safety issues. Drone detection plays a vital role in guaranteeing safety, privacy protection, airspace management, and infrastructure protection.

Currently, common technologies for drone detection include radar, acoustic sensors, and radio frequency (RF) sensors. However, these methods have certain drawbacks [2]. Specifically, radar equipment is costly and has difficulty accurately detecting small and intruding drones [3]. Acoustic

sensors pose challenges in drone detection, particularly when struggling to detect distant drones in noisy environments [4]. RF sensors are vulnerable to unauthorized reading and malicious tampering. In contrast, drone detection based on vision is appealing. Visual techniques are widely used for object detection.

Existing visual detection methods commonly rely on deep neural networks (DNNs) [5], [6]. The DNN detection approaches encompass two- and single-stage techniques. Two-stage detection methods employ region proposal networks (RPNs) at the outset to produce potential object proposals, which are refined and classified in the subsequent stage. The multiple steps involved in the two-stage detection process make it slower compared to single-stage techniques. Typical single-stage detection strategies include the YOLO series [7], [8], [9], [10], [11], [12], [13], [14], [15], the SDD series [16], anchor-free algorithms [17], [18], and anchor-based algorithms [19], [20]. The single-stage detection strategy is based on regression, directly predicting category confidence and locating target positions in images. Nonetheless, it is less effective for small objects and objects of varying scales. The diverse shapes and designs of drones, combined with their rapid movements against complex backgrounds, further complicate the detection process. The You Only Look Once (YOLO) series models have proven useful for drone detection, but they neglect to capture local and global dependencies within and across features. Existing drone detection research has several areas for improvement, such as:

- **Feature Extraction:** Current methods cannot simultaneously capture fine-grained details and global information efficiently. There is a gap in accurately and comprehensively extracting and fusing features in current models. Existing models also lack effective mechanisms to capture local and global dependencies in information, which are essential for accurate drone detection.
- **Computational Complexity:** Current DNN modeling methods frequently introduce excessive computational complexity to increase drone detection accuracy. The computational complexity of DNNs can be accelerated through several methods. This suggests that current drone detection models should have their computational efficiency increased.
- **Attention Distribution Challenge:** It is still a challenge to handle outlier values in the attention distribution of existing drone detection models. Attention should be focused on the most relevant regions of features. This will help improve the overall effectiveness and robustness of the models. Therefore, new schemes are needed to calculate attention weights and distribute attention adequately across significant regions.

In response to these challenges, we design feature capture and modeling modules combined with YOLOv8 [12] for drone detection. The feature capture module (FCM) leverages local attention to selectively focus on a specific region of

drone features based on [6], enabling the model to prioritize relevant information. Additionally, the FCM incorporates global attention to consider information from the entire feature set, capturing global and long-range dependencies among features. Then, the model can fully comprehend the features and capture significant relationships across them. The feature modeling module (FMM) is designed with simple mathematical operations to adjust attention weights relying on the principle [21], eliminating the need for parameter optimization. Furthermore, it enhances the essential regions, facilitating the real-time inference of the decision-making process. The main contribution is summarized as follows:

- We introduce the feature capture module (FCM). The FCM comprises two branches to capture fine-grained details and global information from features. The first branch analyzes local dependencies within the feature sets and interprets the fine-grained details inherent in features. The second branch extends the analysis to encompass broader relationships and capture global dependencies across the entire features. In conclusion, our dual-branch FCM presents a more nuanced and comprehensive approach to feature analysis, leading to superior drone detection compared to existing single-branch or less integrated methods in the literature.
- We provide a FMM and an energy function for feature modeling. Our FMM recalibrates features in a channel-wise manner. The energy function leverages channel-wise statistics to selectively focus on relevant features, improving the modeling of more discriminative features. Because the modeling module introduces no parameters, it contributes to reducing computational complexity. In contrast, many existing modeling methods introduce excessive computational complexity or rely on additional parameters.
- We introduce a compiler to optimize the proposed drone detection models according to the principle [22]. The inter- and intra-operator scheduling are unified by task abstraction. We make scheduling decisions at compile time and allocate the tasks derived from the data flow graph to appropriate hardware components efficiently. This operation reduces runtime overhead and abstracts hardware accelerators as virtualized parallel devices for efficient scheduling. The optimization increases hardware utilization efficiency and decreases inference time compared with others.

Our FCM detects drones accurately with its dual-branch approach to grasp local and global information about drones. This precise detection capability addresses the challenge of drone misuse in restricted areas. Our modules also help in the timely identification of potential drone threats to public safety by distinguishing drones from other objects or animals. Our experiments on two datasets have demonstrated the efficacy of our methods in improving drone detection. Specifically, the FCM and FMM result in a 4.95% and 0.98% increase in  $AP_{0.50:0.95}$  on DUT-Anti-UAV, respectively. Our inference speed is at least 92.84% higher than before by introducing a

**TABLE 1.** Summary of the advantages and disadvantages of different detection algorithms.

Algorithms	Advantages	Disadvantages
Single-stage detectors (e.g., YOLOv2-v9)	<ul style="list-style-type: none"> <li>• Real-time detection speed</li> <li>• Effective on fast-moving objects</li> </ul>	<ul style="list-style-type: none"> <li>• May sacrifice accuracy for speed</li> <li>• Limited in capturing intricate features</li> </ul>
Other single-stage detectors (e.g., FCOS, RetinaNet, SSD, EfficientDet)	<ul style="list-style-type: none"> <li>• Avoid laborious calculations associated with anchor boxes</li> <li>• Fast training and testing</li> <li>• Simple model structure</li> </ul>	<ul style="list-style-type: none"> <li>• May struggle with accuracy, especially for small objects</li> <li>• Not ideal for the detection of dense targets or partially obscured targets.</li> </ul>
Two-stage detectors (e.g., R-CNN, Faster R-CNN, Cascade R-CNN)	<ul style="list-style-type: none"> <li>• Known for high accuracy</li> <li>• Can handle complex scenes and various object sizes</li> </ul>	<ul style="list-style-type: none"> <li>• Memory-intensive</li> <li>• Higher parameter and FLOPs compared to single-stage detectors</li> </ul>
Other methods (e.g., GN + WS, PointRCNN, DETR, VoxelNet, ViT-FRCNN, RetinaNet)	<ul style="list-style-type: none"> <li>• Improved detection accuracy and efficiency in challenging environments, such as dynamic scenes or in the presence of small or fast-moving objects.</li> </ul>	<ul style="list-style-type: none"> <li>• Increased complexity leading to higher computational demands</li> <li>• Limitations in deployment on hardware-restricted devices</li> </ul>

compiler. This is crucial for quickly responding to potential security threats posed by drones.

## II. RELATED WORK

This section provides an extensive overview of recent advancements in drone detection technology.

### A. REAL-TIME DRONE DETECTION ALGORITHMS RELYING ON YOLO

Recently, the single-stage algorithms represented by YOLO have evolved rapidly. Table 1 summarizes the advantages and disadvantages of classic target detection methods. The YOLO series [7], [8], [9], [10], [11], [12], [13], [14] emphasized real-time drone detection. They achieve remarkable speed with their single forward pass for both classification and localization. For instance, Misbah et al. [7] developed a deep learning-based TinyFeatureNet (TF-Net) using infrared imaging, improving upon YOLOv5s architecture [8] for night-time drone detection. Khan et al. [9] integrated transfer learning with YOLOv5 and YOLOv7 to enhance drone detection performance. Li et al. [11] modified YOLOv8 [12] for drone detection. They enhanced feature fusion, reduced parameter costs with Ghostblock V2, and improved bounding box regression with WiseIoU loss. YOLOX [13] introduced a YOLO detector that adopts an anchor-free approach. It integrates the SimOTA label assignment approach with decoupled heads. PP-YOLOE [14] and PP-YOLOE+ [15] incorporated a robust backbone, an anchor-free paradigm, and a neck for object detection. While these YOLO-based models are designed for speed, they do not adequately emphasize the integration of both local and detailed features with broader contextual information. In contrast to these YOLO-based algorithms, our model integrates fine-grained

details and global information by designing a FCM. The module comprises dual branches: one for local attention, capturing detailed features, and the other for global attention, capturing broader contextual information. This dual approach capture a more comprehensive features, improving accuracy and robustness in drone detection.

### B. DRONE DETECTION ALGORITHMS OF BALANCING ACCURACY AND EFFICIENCY

Some methods focus on improving the detection accuracy of different object sizes and shapes [16], [17], [18], [19], [20], [23], [24], [25], [26]. Specifically, the single-pass detector, SSD [16], employed multiple feature maps at different resolutions for prediction. FCOS [17] avoided the laborious calculating associated with anchor boxes by removing the predetermined set of anchor boxes. CenterNet [18] modeled objects as single points and used keypoint estimation. After locating center points, CenterNet regressed to all other characteristics of objects. DCNv2 [19] improved modeling power using deformable convolutions. It handles irregular object forms and matches the geometric modifications of objects. RetinaNet [20] used a single-stage detector and focal loss function to imbalance the foreground-background classes.

Ren et al. [23] designed Faster R-CNN. It integrated RPNs to estimate object boundaries and objectness ratings. Cai et al. [24] adopted a multi-stage object detection paradigm (Cascade R-CNN). Their parameters and FLOPs are higher than those of Ren et al. [23]. Chen et al. [25] enhanced multi-scale object detection by constructing a hierarchical feature pyramid, known as FPG. FPG has more parameters and FLOPs than Cascade R-CNN. GN + WS [26] used batch channel normalization and Weight

Standardization to improve micro-batch training for computer vision tasks. These methods [16], [17], [18], [19], [20], [23], [24], [25], [26] are optimized for high detection accuracy. However, they cannot model more discriminative features. Unlike these methods, we model more discriminative features through the use of energy functions and closed-form solutions. They automatically adjust the weights of features with their statistical properties, thereby enhancing relevant features without introducing additional parameters. This approach improves both accuracy and efficiency, providing a balanced solution for drone detection.

### C. DRONE DETECTION ALGORITHMS OF ADDRESSING COMPLEX ENVIRONMENTS CHALLENGES

Some studies have enhanced the precision and effectiveness for drone detection under different conditions. This includes dealing with dynamic backgrounds [27], distinguishing drones from other objects like birds [28], and improving the detection of small or distant drones [29]. Seidaliyeva et al. [27] segmented the task into two parts: background subtraction to identify moving objects and CNN classification to categorize objects. Dong et al. [28] proposed an enhanced SSD (single-shot multibox detector) method. Wang et al. [29] detected small, distant drones with limited pixel area and morphological features in video streams. They introduced a drone detection method with feature super-resolution and motion information extraction. While recent DNN models have achieved comparable accuracy, they are still large and require substantial time for inference. Different from these methods, the proposed method introduces an optimization architecture specifically designed to handle real-time drone detection more effectively. This architecture eliminates redundant operations and streamlines task execution paths, significantly cutting down inference time. By using a compiler to schedule tasks both within and between operators, our method allocates tasks efficiently and enhances hardware utilization. As a result, our approach detects drones faster and more reliably, even in complex environments, while maintaining high accuracy during real-time operation.

## III. METHOD

### A. PIPELINE OF THE PROPOSED ALGORITHM

Our model architecture is shown in Fig. 1. The proposed modules include feature capture, modeling, and optimization modules. They perform feature extraction, attention mechanisms, and weight adjustment prior to the detection phase. They play a vital part in enhancing drone detection. The optimizing compiler improves our execution speed by reducing redundant operations in the code execution path.

The pipeline begins with a ‘Conv’ block specified with kernel size ( $k = 3$ ), stride ( $s = 2$ ), and padding ( $p = 1$ ), indicating a convolution operation. Following this is a repeated structure labeled ‘4(Conv + C2f)’-suggesting there are four instances of a sequential operation consisting

of a convolution (‘Conv’) followed by a module ‘C2f’ in YOLOv8 [12]. These backbone structures encompass the feature extraction architecture derived from YOLOv8. The four stacked operations represent a deep feature extraction phase where multiple levels of abstraction are obtained. The ‘SPPF’ block is a variant of spatial pyramid pooling. It pools the feature maps at different scales to maintain spatial information at various resolutions.

Further, to create a rich, multi-scale feature representation, we employ the upsampling (‘U’) and concatenation (‘C’) operations based on YOLOv8. The lower-resolution feature is upsampled to match the resolution of the higher levels. The upsampled features are concatenated with the corresponding higher-resolution features. This process enhances the accuracy by combining the context from lower levels with the detail from higher levels. Then we perform C2f, convolution, and feature map splicing. The SPPF block and two C2f output three feature maps. The outputs are used as the inputs for our FCM. The outputs of the FCM is adopted as the inputs of our FMM. The features outputted by the FMM acted as the inputs of the Conv modules.

In the FCM, we apply convolution, batch normalization (Bn), and ReLU activation functions following [6]. We integrate the information of adjacent features through weighted summation. Additionally, the input feature  $\alpha$  passes through ‘Adaptive average pooling,’ followed by ‘Feature computing’ and activation function ‘Bn + ReLU’. We then add the results of the local and global attention branches. This step is to combine local features and global context information. We map the weighted feature map to the (0, 1) interval to serve as the attention weight. The original input feature map  $\alpha$  and the attention weight are element-wise multiplied to produce the final weighted feature map. This process strengthens the focus on important features while suppressing unimportant information, allowing the network to focus more on features that are beneficial to drone detection.

In the FMM, the features are fed into the proposed ‘Energy function’. This function is involved in optimization, ‘Minimization’ and obtaining a ‘Closed-form solution’. We use this solution to ‘Adjust weights’ of the features following [21]. This weight is based on the statistical properties of the feature and takes into account the impact of feature variation, with the aim of determining the relative importance of each feature in the new feature map. Then, through ReLU and element-wise multiplication, the features are updated.

The block labeled ‘Conv + C’ denotes a convolutional layer followed by a concatenation operation. The outputs of the FMM are used the inputs of the concatenation operation in ‘Conv + C’. The convolutional layer is used for feature extraction from the inputs. Following the ‘Conv + C’ block, there is a ‘C2f’ block. Next is another ‘Conv + C’ block with the same parameter setting as before, indicating a repetition of the convolution and concatenation process to further refine the feature maps. Afterward, we adopt another ‘C2f’ block. The detector in YOLOv8 is the final stage in the model, where

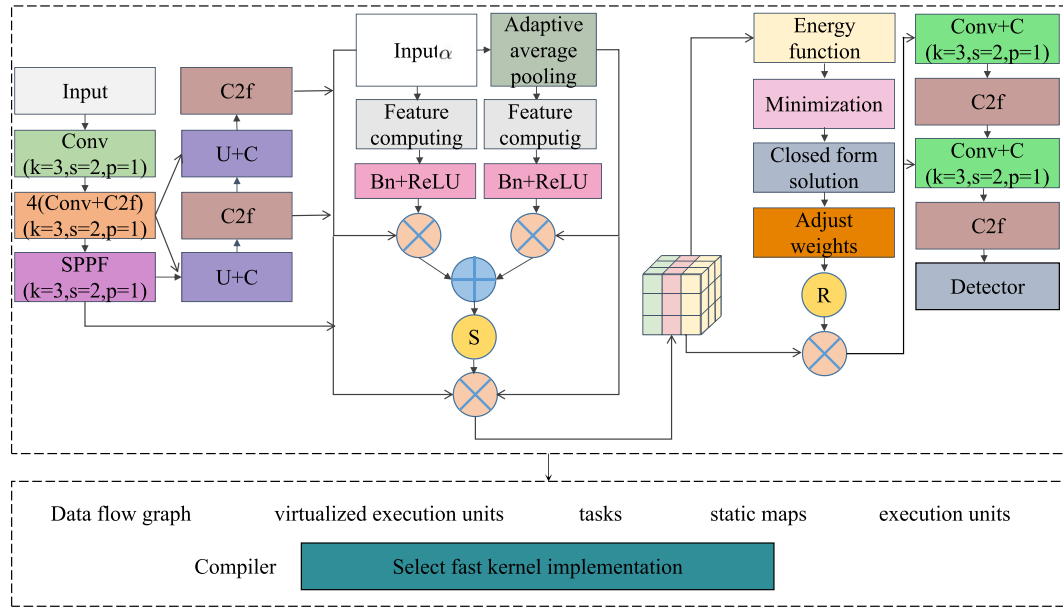


FIGURE 1. Overview of the proposed feature processing, capture, modeling, and optimization techniques for real-time drone detection.

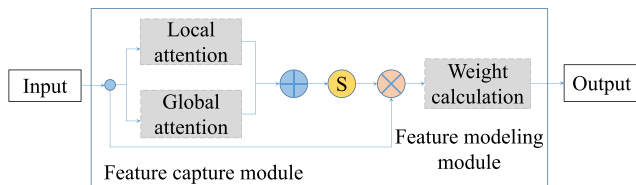


FIGURE 2. Feature capture and modeling modules using local and global attention mechanisms.

the processed features are used to identify and localize drones within the original input.

This drone detection model serves as input to our compiler. We optimize drone detection models using TorchScript, a data flow graph (DFG), virtualized execution units (vEUs), tasks, static maps, and execution units (EUs) based on [22]. The drone detection models are serialized in a format called TorchScript. The DFG helps understand dependencies, optimize processes, and ensure efficient task execution. The vEUs are used to plan and optimize tasks without relying on physical hardware. Tasks are processed individually and scheduled on the hardware in an efficient order. Static maps provide a predetermined sequence for task execution, ensuring predictability. Tasks are converted into machine code that can be executed directly on hardware components called execution units (EUs). These optimization steps aim to deploy accurate and real-time drone detection on hardware while maximizing hardware utilization.

Our contribution resides in the proposed modules and the optimization of the drone detection model. Primarily, we design the FCM to extract fine-grained details and global information from features. This facilitates a more sophisticated and comprehensive fusion of features. This point is

often not emphasized enough in previous studies. Secondly, we introduce the FMM to accentuate significant regions and suppress irrelevant information. This empowers the modeling of more discriminative features. By designing energy functions and closed-form solutions to adjust feature weights, we can dynamically adjust the relative importance of features in new features based on their statistical properties. This innovation was less common in previous studies. Thirdly, we provide a compiler to optimize the proposed model. The proposed model is optimized by fine-grained compilation, leading to reduced scheduling overhead. In short, we improve the accuracy and efficiency of drone detection through advanced feature processing and optimization technology, especially the comprehensive innovation in feature capture, modeling and system optimization.

### B. FEATURE CAPTURE MODULE

The FCM includes local and global attention branches based on the method [6], as shown in Fig. 2. A local attention branch is implemented to capture local features, as shown in Step 1 of Algorithm 1. Specifically, channels are reduced through 2D convolutions, where  $\beta_\lambda$  refers to a batch size. The dimension of  $\beta_\lambda$  is 12. The operation used is cross-correlation, denoted by  $\star$ . The output and input values are represented by  $d(\beta_\lambda, N_{v_\mu})$ , and  $in(\beta_\lambda, v)$ , respectively. The input and output channel indexes are  $v$  and  $\mu$ , respectively. The output and input channel numbers are represented by  $N_v$  and  $L_\lambda$ , respectively. The term  $f(N_{v_\mu}, v)$  represents a filter. The term  $b(N_{v_\mu})$  represents a learnable bias. The dimensions of  $L_\lambda$  and  $N_v$  are  $(\beta_\lambda, v, H, W)$  and  $(\beta_\lambda, \mu, H, W)$ .  $H$  and  $W$  are the height and width of features in pixels, respectively. In the local attention branch, the convolutions are used to

capture local patterns and dependencies. They contain an  $1 \times 1$  kernel with one stride and no padding. The kernel size of  $1 \times 1$  means that the local attention branch primarily focuses on channel-wise dependencies. We reduce the number of channels in the convolutions and decrease their complexity.

Subsequently, the outputs are normalized using batch normalization (BatchNorm2d). This normalization process aids in stabilizing the training procedure and enhancing the model generalization capabilities. Following BatchNorm2d, the element-wise ReLU is used. Subsequently, another 2D convolutional layer, similar to the previous convolution, is employed to restore the output to its original number of channels. Finally, BatchNorm2d is again applied to normalize the output of the second convolution. Overall, the local attention branch captures local spatial dependencies through channel-wise transformation and feature map normalization.

---

#### Algorithm 1 Feature Capture Module

---

**Require:** a feature map  $\alpha$

**Ensure:** a feature map  $\beta_1$  after attention weighting

- 1: Define local attention branch:
    - $d(\beta_\lambda, N_{v_\mu}) = b(N_{v_\mu}) + \sum_{v=0}^{L_\lambda-1} f(N_{v_\mu}, v) \star in(\beta_\lambda, v)$ ;
    - Apply batch normalization after each convolution;
    - Apply ReLU activation after each batch normalization;
  - 2: Apply local attention branch to  $\alpha$  and store the result as  $xa$ ;
  - 3: Define global attention branch:
    - Apply adaptive average pooling;
    - $d(\beta_\lambda, N_{v_\mu}) = b(N_{v_\mu}) + \sum_{v=0}^{L_\lambda-1} f(N_{v_\mu}, v) \star in(\beta_\lambda, v)$ ;
    - Apply batch normalization after each convolution;
    - Apply ReLU activation after each batch normalization;
  - 4: Apply global attention branch to  $\alpha$  and store the result as  $xb$ ;
  - 5:  $xab = xa + xb$ ;
  - 6:  $sxab = \sigma(xab)$ ;
  - 7:  $xx = \alpha \odot sxab$ ;
- 

In Step 2, we utilize the local attention branch for the input  $\alpha$  and save the outcome as  $xa$ . The local attention branch adeptly identifies short-range dependencies by focusing exclusively on adjacent features. In Step 3, a global attention branch is designed to capture features. This global block comprises several layers, including two convolutional layers, adaptive average pooling, and two BatchNorm2d layers. The adaptive average pooling aggregates and compresses global information. It captures global context in inputs. It retains channel-wise information, giving a holistic view of the entire feature map by averaging spatial dimensions. Then, the feature spatial dimensions are reduced to  $1 \times 1$ . Through the convolutional layer, the feature map channel is reduced. Two convolutions are applied, which include an  $1 \times 1$  kernel, a stride of 1, and no padding. This step aims to extract features and decrease computational complexity. After each convolution, BatchNorm2d is applied to normalize feature maps.

After each convolution, the feature map is subjected to a ReLU activation function. The subsequent operations in the global attention branch operate on a much smaller spatial size, making it computationally efficient. The reason is that the global attention branch reduces the spatial dimensions early in features. By utilizing this global attention branch, we capture the global information in feature maps, facilitating improved comprehension of the overall features. In Step 4, our global attention branch is applied to  $\alpha$ , and the resulting value is stored as  $xb$ . In Step 5, we calculate the sum of  $xa$  and  $xb$  and store the sum as  $xab$ . In Step 6, we apply the sigmoid activation function to  $xab$  and store the result as  $sxab$ . Finally, we multiply element-wise  $\alpha$  with  $sxab$  and store the result as  $xx$  in Step 7.

In practice, combining local and global attention is beneficial, as they capture different levels of context and details, providing a more comprehensive understanding of the input. The local and global attention modules play pivotal roles in capturing local dependencies within the features and global dependencies across the entire feature set, respectively. The global attention module assists in capturing significant global features. The local attention module enables more detailed analysis and precise feature capture. By combining these two approaches, models can proficiently extract pertinent features, leading to improved performance in drone detection.

#### C. FEATURE MODELING MODULE

In this section, to achieve better performance, we present a simplified FMM that does not require parameters. The module is illustrated in Algorithm 2. Our algorithm leverages channel-wise statistics to calculate attention maps. It enhances the representational capacity of the FCM by selectively focusing on relevant features. Specifically, we leverage the principle of [21] and propose an energy function to determine the importance of features as follows:

$$E_o(\text{weight}_o, \text{bias}_o, \chi, Y_{in}) = \frac{1}{N} \sum_{j=1}^N (\iota - \mathbb{1}_o Y_j - \text{bias}_o)^2 + (\kappa - \mathbb{1}_o \chi - \text{bias}_o)^2, \quad (1)$$

where  $\iota$  and  $\kappa$  are binary labels. The target neuron is denoted as  $\chi$ .  $Y_{in}$  is the other neuron in one channel of the input feature. The spatial dimension index is denoted by  $in$  ( $in \in (0, N)$ ), where  $N$  represents the overall count of neurons in a single channel. The  $\text{bias}_o$  denotes bias. The  $\mathbb{1}_o$  signifies a weight. Eq. (1) calculates the attention weights for a given feature map. It assesses the similarity between the feature vector of a neuron and those of all others within the same channel.

The input of Algorithm 2 is the feature  $\beta_1$ , which is the output of Algorithm 1. The dimensions of  $\beta_1$  vary based on the input of the entire model. A lower value of  $E_o$  indicates that the neuron is more distinct from its neighbors and thus holds greater significance in feature modeling. In Step 1 of Algorithm 2, we solve Equation (1) and acquire the

closed-form solution  $c_o$ . We compute the variance  $\varphi_{(o)}^2$  in Step 2. In Step 3, we calculate the weight  $(\psi_{(o)}/2\varphi_{(o)})^2$  with  $c_o$ . The  $\psi_{(o)}$  is equivalent to the target neuron  $o$  minus the mean  $v_{(o)}$ , namely  $\frac{1}{M} \sum_{in=1}^M Y_{in}$ . The feature  $\beta_1$  is multiplied by the weight  $R((2\varphi_{(o)}/\psi_{(o)})^2)$ , resulting in the creation of  $\beta_2$  in Step 4. The dimensions of  $\beta_2$  and  $\beta_1$  are identical.

---

**Algorithm 2** Feature Modeling Module
 

---

**Require:** a feature map  $\beta_1$

**Ensure:** a new feature map  $\beta_2$

- 1: Obtain the closed-form solution  $c_o = \frac{(\circ+v_{(o)})\psi_{(o)}}{(\psi_{(o)})^2+2\varphi_{(o)}^2}$  according to Equation (1)
  - 2:  $\varphi_{(o)}^2 = \frac{1}{M} \sum_{in=1}^M \left( Y_{in} - \frac{1}{M} \sum_{in=1}^M Y_{in} \right)^2$
  - 3: Calculate the weight  $(\psi_{(o)}/2\varphi_{(o)})^2$  based on  $c_o$ .
  - 4:  $\beta_2 = \beta_1 \odot R((2\varphi_{(o)}/\psi_{(o)})^2)$
- 

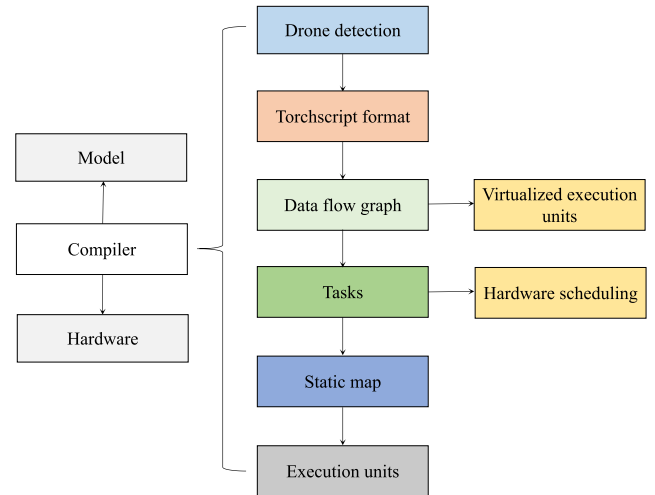
Through Algorithm 2, we use a scaled inverse multiplicative function to compute attention weights. This approach reduces the impact of outliers and extreme values on the attention distribution. The more distinctive the feature is from the surrounding ones, the more important it is for drone detection. Thus, the proposed model effectively model the most important information without extra parameters.

#### D. MODEL OPTIMIZATION

We optimize the drone detection models relying on [22], as depicted in Fig. 3. Before further processing, we convert the drone detection model into a TorchScript model. TorchScript is a serialized format for PyTorch models. This step aims to optimize the model and integrate it with hardware systems. Then, the TorchScript-formatted model is translated into a data flow graph (DFG). The DFG represents the sequence and the parallel operations that occur during computation [22]. We adopt the DFG format to understand dependencies within computations, optimize processes, and ensure that tasks are executed in an efficient order.

The virtualized execution units (vEUs) act as representations for the actual hardware components. By using vEUs, we plan and optimize drone detection tasks without directly interfacing with the physical hardware. This allows for the tasks to be scheduled and processed without immediate reliance on the physical hardware. Therefore, we enhance the flexibility and scalability of compiling drone detection models. We process the individual operations or instructions derived from the DFG, which are tasks. These are the atomic units of execution in the system.

Then, we perform hardware scheduling. This step determines the order and priority of task execution on the actual hardware. It ensures that tasks are allocated to the appropriate hardware components in a manner that maximizes efficiency. The tasks play a pivotal role in the processing sequence. As broken-down units of larger operations, they represent granular computational steps. These tasks are



**FIGURE 3.** Optimization of drone detection models through TorchScript conversion, data flow graph analysis, and virtualized execution unit scheduling for efficient hardware integration.

not only dependent on the DFG but are also closely integrated with the vEUs. Their symbiotic relationship ensures that tasks are executed in the most efficient manner, maximizing computational speed and minimizing resource consumption.

We use the static map to execute tasks. The static maps are fixed representations. They are unlike dynamic mapping, which changes in real time. They provide a predetermined sequence for task execution. This ensures a certain degree of predictability and consistency. The operation is especially beneficial when handling complex operations like drone detection modes that require structured execution. We convert the tasks on each static graph into machine code that can be executed directly on execution units (EUs). The EUs are the actual hardware components that execute the tasks. Each execution unit takes on a task, processes it as per the instructions, and then moves on to the next one. The high-level instructions are translated into a format that the hardware can understand and execute. The drone detection models are optimized by reducing scheduling overheads and maximizing hardware utilization.

The above workflow constitutes the basic optimization steps of the compiler. The end goal of the optimization is not just processing but efficient deployment on various edge devices for drone detection. The compiler translates tasks into machine codes compatible with devices. It ensures that drone detection is not only accurate but also rapid and real-time. At the same time, we encapsulate the intricacies of data flow, task management, and real-time execution. By using the above optimization methods, we achieve seamless integration of software models with physical hardware.

#### IV. EXPERIMENT INFORMATION

In the paper, the DUT-Anti-UAV [30] and Online Drone [31] datasets are used to train and assess drone detection models.

### A. DATASET

- The DUT-Anti-UAV dataset consists of detection and tracking parts. The detection part has 5,200 training images, 2,600 validation images, and 2,200 testing images. The dataset covers more than 35 different types of miniature drones. They occur in diverse outdoor environments, including various lighting conditions and different weather conditions. Drones appear in the sky, trees, buildings, farmland, playgrounds, and other rich variety of objects and complex backgrounds. The significant outdoor lighting changes in the dataset help prevent model overfitting and are critical for training robust drone detection models.
- The Online Drone dataset includes large and small drones. The dataset has 1,433 training images and 359 testing images. The dataset incorporates many different drones and diverse scene information, such as buildings and jungles. Chen et al. [31] performed data augmentation on this dataset, including image quality enhancement, lighting changes, and geometric transformation. From this, the foreground drone is processed by a blur filter to blur it. The drones have shadows. Standard shadow maps are created by random lines. Irregular shadow maps are generated by Perlin noise. The foreground drones are evenly scaled horizontally and vertically. The drone position has different variations in the background image. These operations add to the complexity and robustness of the dataset. Therefore, we use the data-augmented dataset.

The above two datasets have reliable annotation quality, diverse targets, and diverse scenarios. Their labels are accurate, complete, and consistent. These drone targets have rich changes in size, shape, attitude, etc. Both datasets contain diverse scenes, covering different environmental conditions, lighting conditions, background complexity, and other factors. These datasets have been widely used by researchers [1], [5], [30], [31] to evaluate model performance.

### B. EVALUATION METRICS

The following metrics are used to validate our algorithm:

- Average recall (AR): the AR metric used in this paper includes  $AR_{m1}$ ,  $AR_S$ ,  $AR_M$ , and  $AR_L$ .  $AR_{m1}$  represents the  $AR_{0.50:0.95}$  given one detection per image.  $AR_{0.50:0.95}$  denotes AR averaged across ten IoU thresholds: (0.5: 0.05: 0.95). Here, the IoU threshold quantifies the localization of drone detection, and its setting is dynamic.  $AR_M$  represents  $AR_{0.50:0.95}$  for medium-sized objects with areas ranging from  $32^2$  to  $96^2$ .  $AR_S$  corresponds to the  $AR_{0.50:0.95}$  for small objects in areas below  $32^2$ .  $AR_L$  denotes  $AR_{0.50:0.95}$  for big objects of areas exceeding  $96^2$ .
- Average precision (AP): the AP metric used in this paper includes  $AP_{0.50:0.95}$ ,  $AP_{0.50}$ ,  $AP_{0.75}$ ,  $AP_S$ ,  $AP_M$ , and  $AP_L$ .  $AP_{0.50:0.95}$  denotes AP averaged over ten IoU thresholds: (0.5: 0.05: 0.95).  $AP_{0.50}$  represents AP at

**TABLE 2.** Experiment parameters and hardware configuration for training drone detection models using PyTorch on Ubuntu operating system.

Items	Parameters
Programming languages	Python 3.8.16
Epoch	24
PyTorch	1.12.1
Input shape	$640 \times 640$
Batch size	12
Output shape	$640 \times 640$
Weight decay	$5 \times 10^{-4}$
Momentum	$9.37 \times 10^{-1}$
Learning rate	$10^{-3}$
Optimizer	SGD
Memory	64 GB
Scheduler	Linear
CPU	Intel i9-9900K
GPU	NVIDIA RTX 2080 Ti
Operating systems	Ubuntu 18.04.5

the 0.50 IoU threshold.  $AP_{0.75}$  is AP at the 0.75 IoU threshold.  $AP_S$  is  $AP_{0.50:0.95}$  for small objects with areas below  $32^2$ .  $AP_L$  is  $AP_{0.50:0.95}$  for big objects of areas exceeding  $96^2$ .  $AP_M$  is  $AP_{0.50:0.95}$  for medium-sized objects with areas ranging from  $32^2$  to  $96^2$ .

- Model and computational complexity: the model and computational complexity are measured by parameter quantities and FLOPs, respectively. The more parameters, the higher the model complexity. The bigger the FLOPs, the lower the computational complexity.
- Average frames per second (FPS): FPS signifies the amount of frames handled per second. The FPS mirrors the processing speed of algorithms. The lower the FPS, the slower the speed. We evaluate the average FPS of different algorithms using 2,000 images. These evaluation metrics are widely used in various fields.

### C. EXPERIMENT SETUP

We set up the same experimental environment for all algorithms. Table 2 provides details about the experimental setup, including the software and hardware, as well as the configuration of the training process. The Python 3.8.16 and PyTorch 1.12.1 were used. Our model training resources are limited. The epoch can only be designed to be small. We found the results are better when the epoch is 24 than those when the epoch is 8 or 16 through experiments. Therefore, our training process involved 24 epochs. The initial learning rate was  $10^{-3}$ . The momentum value used during training was  $9.37 \times 10^{-1}$ . The experiment was conducted on Ubuntu 18.04.5, which provides 24 training epochs with the same configurations.

### V. RESULT

In object detection, there are one- and two-stage CNN-based strategies according to different anchor generation



**TABLE 3. Comparative performance of baseline and proposed methods on online drone and DUT-Anti-UAV datasets across various metrics.**

Dataset	Method	$AP_{0.50:0.95}$	$AP_{0.50}$	$AP_{0.75}$	$AP_S$	$AP_M$	$AP_L$	$AR_{m1}$	$AR_S$	$AR_M$	$AR_L$
Online Drone	baseline [12]	0.461	0.884	0.399	0.107	0.426	0.559	0.508	0.232	0.527	0.643
Online Drone	+Algorithm 1	0.463	0.889	0.407	0.114	0.438	0.569	0.509	0.232	0.529	0.643
Online Drone	+Algorithm 1+2	0.473	0.894	0.447	0.132	0.466	0.584	0.515	0.239	0.550	0.651
DUT-Anti-UAV	baseline [12]	0.576	0.869	0.648	0.434	0.651	0.734	0.605	0.517	0.710	0.796
DUT-Anti-UAV	+Algorithm 1	0.598	0.882	0.676	0.455	0.654	0.735	0.636	0.521	0.712	0.796
DUT-Anti-UAV	+Algorithm 1+2	0.612	0.913	0.693	0.455	0.665	0.739	0.639	0.532	0.715	0.797

**TABLE 4. Comparison of detection models on the DUT-Anti-UAV dataset.**

Method	$AP_{0.50:0.95}$	$AP_{0.50}$	$AP_{0.75}$	$AP_S$	$AP_M$	$AP_L$	$AR_{m1}$	$AR_S$	$AR_M$	$AR_L$
YOLOX [13]	0.521	0.901	0.556	0.466	0.542	0.562	0.584	0.531	0.600	0.631
PP-YOLOE [14]	0.544	0.830	0.635	0.473	0.585	0.579	0.570	0.519	0.687	0.704
SSD300 [16]	0.524	0.875	0.558	0.375	0.528	0.696	0.592	0.473	0.598	0.752
FCOS [17]	0.363	0.849	0.193	0.376	0.376	0.320	0.475	0.460	0.490	0.473
CenterNet [18]	0.251	0.377	0.298	0.578	0.691	0.154	0.713	0.668	0.742	0.731
DCNv2 [19]	0.621	0.944	0.714	0.550	0.664	0.651	0.671	0.604	0.705	0.712
RetinaNet [20]	0.583	0.951	0.663	0.477	0.615	0.647	0.653	0.588	0.678	0.706
Faster R-CNN [23]	0.635	0.945	0.755	0.552	0.671	0.687	0.683	0.605	0.716	0.740
Cascade R-CNN [24]	0.574	0.903	0.652	0.480	0.618	0.625	0.630	0.553	0.663	0.687
FPG [25]	0.283	0.511	0.296	0.014	0.393	0.495	0.343	0.017	0.480	0.586
GN + WS [26]	0.504	0.893	0.528	0.456	0.542	0.507	0.574	0.538	0.600	0.583
Ours	0.612	0.913	0.693	0.455	0.665	0.739	0.639	0.532	0.715	0.797

mechanisms. We select the mainstream one- and two-stage object detection models [13], [14], [16], [17], [18], [19], [20], [23], [24], [25], [26] for comparison on two benchmarks. These contrasting algorithms are more in line with actual engineering scenarios and have been published in many papers.

#### A. ABLATION STUDY

The drone detection results improve when our modules are adopted. As demonstrated in Table 3, the baseline attains an  $AP_{0.50:0.95}$  of 0.461 on the Online Drone dataset. The baseline method referenced in this table corresponds to the standard implementation of YOLOv8 in the MMYOLO toolbox [32]. It serves as a foundational comparison point without any of our proposed modifications. When Algorithm 1 is added to the baseline, the  $AP_{0.50:0.95}$  value increases to 0.463. Incorporating both Algorithm 1 and 2, the  $AP_{0.50:0.95}$  data reaches 0.473. The results for  $AP_{0.50}$ ,  $AP_M$ ,  $AP_L$ ,  $AP_S$ ,  $AR_{m1}$ ,  $AP_{0.75}$ ,  $AR_M$ ,  $AR_S$ , and  $AR_L$  are better when Algorithm 1 is used. When Algorithm 1 and 2 are leveraged, the results are further improved. The results on the DUT-Anti-UAV dataset also demonstrate Algorithm 1 and 2 are effective. Importantly, the numbers of FLOPs and parameters of Algorithm 1 are 0.16 G and 0.35 M. Algorithm 2 has no parameters. These results suggest that the proposed modules have a positive impact on accuracy with few parameters and FLOPs.

#### B. EVALUATION OF DETECTION PERFORMANCE

To assess model performance, we calculate the detection accuracy, parameter quantities, FLOPs, and average FPS.

Evaluation of Detection Accuracy on DUT-Anti-UAV: As shown in Table 4, on DUT-Anti-UAV, our method obtains better accuracy than others. Compared with Faster R-CNN, our method has 3.62% lower  $AP_{0.50:0.95}$ . The reason for this outcome is that Faster R-CNN makes use of region proposals, which frequently leads to greater precision. However, by skipping the proposal-generating stage, our method is faster and still produces competitive results. Our method achieves 17.47% and 12.50% higher  $AP_{0.50:0.95}$  than the one-stage lightweight models [13], [14], respectively. Our method has 68.60% better  $AP_{0.50:0.95}$  than the classic framework [17]. The  $AP_{0.50}$  of our algorithm is 0.913, which is 10.00% higher than the 0.830 of the latest lightweight model [14]. The  $AP_{0.50}$  of our algorithm is around 1.11-142.17% higher than that of other lightweight methods. The  $AP_{0.75}$  of our method is 0.693, which is 259.07% higher than the 0.193 of the flexible strategy [17].

Additionally, our method has 9.13-24.64%  $AP_{0.75}$  higher compared with other similar methods [13], [14], [16] in Table 4. Our model also has 21.01%  $AP_S$  compared to the common approach [17]. The  $AP_M$  of our method is 0.665, which is 0.15-76.86% higher compared to that of other studies [13], [14], [16], [17], [19], [20], [24], [25], [26]. Our method has 0.96-47.21%  $AP_L$  higher than comparison studies. In  $AR_{m1}$ , our method surpasses FPG to the greatest extent, reaching 86.30%. Compared with other mainstream methods [13], [14], [17], the  $AR_{m1}$  improvement of our method is between 15.65%, 0.19%, and 2.50%, respectively. Our method yields 9.4%, 19.17%, and 26.31% higher  $AR_S$ ,  $AR_M$ , and  $AR_L$ , respectively, versus the lightweight technique [13]. The proposed method has exceeded current work on most indicators. The reason is that we gather

**TABLE 5.** Comparison of detection models on the online drone dataset.

Method	$AP_{0.50:0.95}$	$AP_{0.50}$	$AP_{0.75}$	$AP_S$	$AP_M$	$AP_L$	$AR_{m1}$	$AR_S$	$AR_M$	$AR_L$
YOLOX [13]	0.428	0.862	0.338	0.136	0.404	0.538	0.494	0.202	0.489	0.597
PP-YOLOE [14]	0.401	0.838	0.295	0.132	0.407	0.488	0.453	0.257	0.579	0.634
SSD300 [16]	0.430	0.863	0.338	0.106	0.389	0.571	0.486	0.150	0.456	0.636
FCOS [17]	0.426	0.902	0.304	0.138	0.421	0.552	0.523	0.180	0.525	0.634
CenterNet [18]	0.371	0.745	0.307	0.083	0.408	0.472	0.481	0.165	0.479	0.589
DCNv2 [19]	0.446	0.894	0.357	0.138	0.435	0.546	0.501	0.178	0.505	0.603
RetinaNet [20]	0.422	0.894	0.312	0.146	0.422	0.522	0.511	0.198	0.515	0.611
Faster R-CNN [23]	0.444	0.894	0.346	0.132	0.421	0.557	0.495	0.178	0.494	0.602
Cascade R-CNN [24]	0.442	0.887	0.372	0.103	0.429	0.553	0.501	0.154	0.499	0.620
FPG [25]	0.367	0.807	0.220	0.053	0.377	0.479	0.449	0.052	0.474	0.550
GN + WS [26]	0.429	0.888	0.328	0.149	0.424	0.532	0.497	0.189	0.495	0.604
Ours	0.473	0.894	0.447	0.132	0.466	0.584	0.515	0.189	0.550	0.651

**TABLE 6.** Comparison of computational efficiency and performance metrics among different detection models on the DUT-Anti-UAV dataset.

Method	FLOPs (G)	Number of parameters (M)	Average FPS
YOLOX [13]	13.32	8.94	129.81
PP-YOLOE [14]	7.92	7.45	78.12
SSD300 [16]	34.27	23.75	100.94
FCOS [17]	206.51	31.84	23.72
CenterNet [18]	13.06	14.21	80.32
DCNv2 [19]	188.15	41.99	20.22
RetinaNet [20]	214.60	36.10	22.91
Faster R-CNN [23]	216.30	41.12	22.43
Cascade R-CNN [24]	244.10	68.93	18.20
FPG [25]	253.70	79.40	25.80
GN + WS [26]	66.96	42.44	15.23
Ours	14.43	11.49	100.51

comprehensive context, capture local connections among features, and adjust attention weights.

Evaluation of Detection Accuracy on Online Drone: Our method attains notably superior performance over others, as shown in Table 5. Specifically, our method yields substantially 6.53% higher  $AP_{0.50:0.95}$  versus Faster R-CNN in  $AP_{0.50:0.95}$ . Our method significantly outperforms CenterNet and FPG by 27.49% and 28.89%, respectively. Compared with YOLOX, FCOS, and RetinaNet, our method shows an improvement of 10.51%, 11.03%, and 12.09%, respectively, in  $AP_{0.50:0.95}$ . In  $AP_{0.50}$ , our model performs similarly to Faster R-CNN, DCNv2, and RetinaNet. However, compared to CenterNet, our model achieves a significant improvement of 20.00%, which is the largest improvement in  $AP_{0.50}$ . Compared to classic models like FPG and YOLOX, our method shows a 3-10% improvement in  $AP_{0.50}$ . Our method achieves a 10.92-103.18% improvement in  $AP_{0.75}$ . In particular, compared to FPG (103.18% improvement), FCOS (47.04% improvement), RetinaNet (43.27% improvement), and CenterNet (45.60% improvement), our method shows an improvement of over 40.00%. Compared to other approaches, our method also exhibits a 20-30% improvement. In  $AP_S$ , our method performs similarly to Faster R-CNN and PP-YOLOE. Especially, our model achieves an improvement of 149.06%, which is the largest improvement. In comparison to Cascade R-CNN and SSD300, our method also shows a 20-30.00% improvement by enhancing discriminative feature modeling.

Compared to CenterNet and FPG methods, our method shows a significant improvement of 23.73% and 21.92%, respectively. Compared to YOLOX and RetinaNet, our suggested technique has a modest drop in  $AP_S$ . This indicates that we have room for improvement to capture more minute features. We will investigate this possibility in further versions. However, our model performs in a well-rounded manner. Especially, it indicates high-quality detections that are in close agreement with annotations from the ground truth, with a competitive  $AP_{0.75}$ . Relative to RetinaNet and YOLOX, our method also exhibits an improvement of 11.88% and 8.55% in  $AP_L$ , respectively. Our method also demonstrates improvements across the majority of metrics. For example, in  $AR_L$ ,  $AP_M$ ,  $AP_S$ ,  $AR_{m1}$ ,  $AR_M$ ,  $AR_S$ , and  $AP_L$ , our model also outperforms most comparison methods on Online Drone. This is because it enables comprehensive analysis of fine-grained and extensive feature relationships.

Evaluation of Model Parameter Amounts, FLOPs, and Average FPS: Our method is real-time with competitive FLOPs and parameters than most high-precision models. As listed in Table 6, our model is contrasted with various approaches in parameter quantities, FLOPs, and average FPS on DUT-Anti-UAV. In the table, the ‘G’ refers to a billion. The ‘M’ refers to a million. The comparison models are implemented by the MMYOLO toolbox [32]. The authors customize the input dimensions of their models.

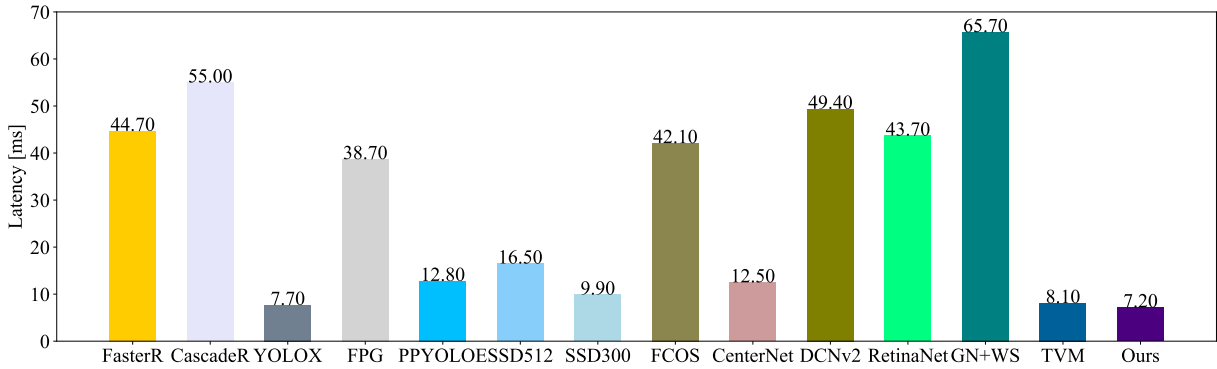


FIGURE 4. Performance analysis of drone detection latency across different models optimized for speed on NVIDIA RTX 2080 Ti GPU.

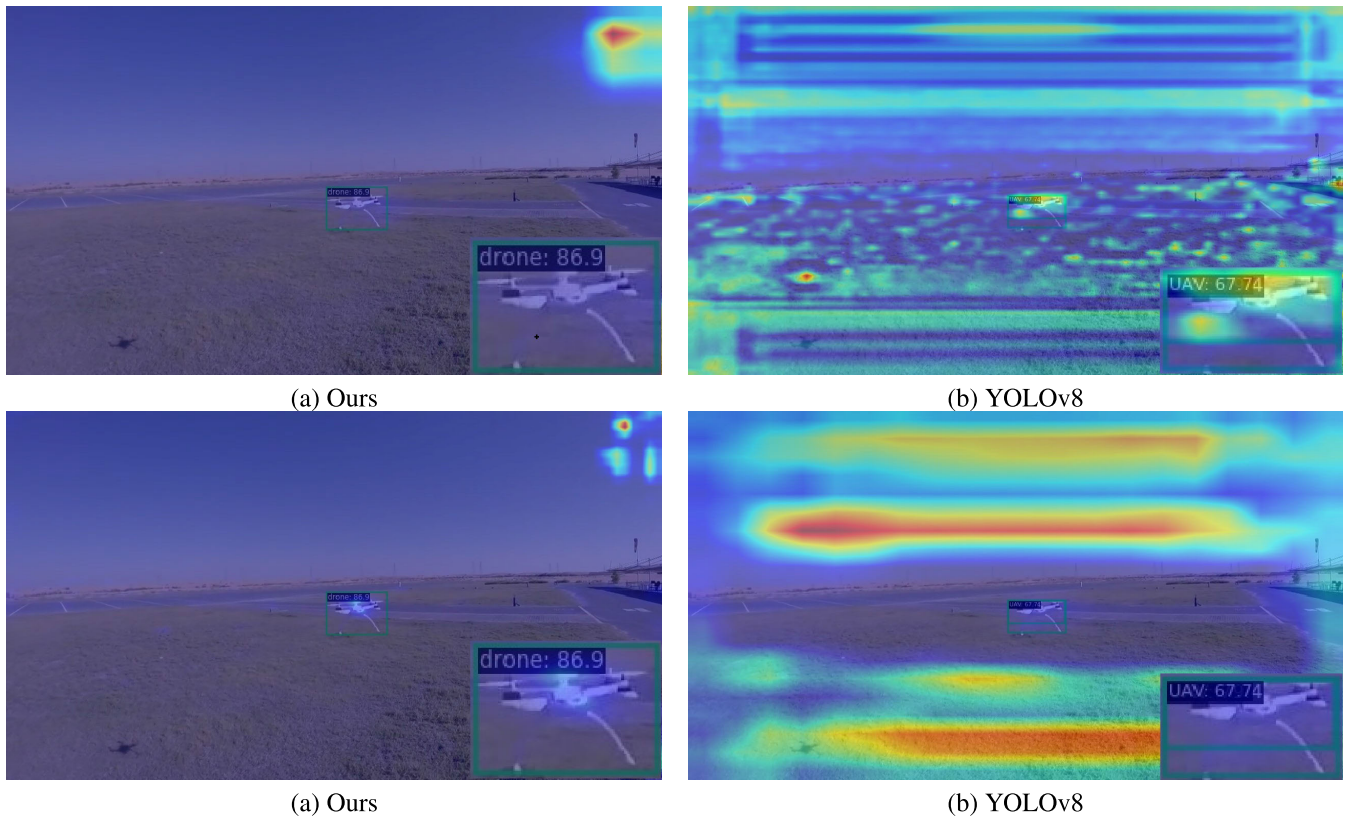


FIGURE 5. Comparative visualization of feature maps outputted by our proposed algorithm refinements and YOLOv8 across different modules.

Compared with other models, our method achieves a more favorable balance in detection accuracy, parameter quantities, FLOPs, and average FPS. For example, our method has 85.53% fewer FLOPs than the high-precision model [25]. Compared with other high-precision models, our method has at least 57.89% fewer FLOPs. Our method has at least 51.62% fewer parameters than other high-precision models [16], [19], [23], [24]. Our average FPS is 100.51, which is less than that of most recent research. This comparison indicates that the suggested approach operates in real time. Our method provides a good balance among the number of parameters, average FPS, and FLOPs. While there are models with fewer

FLOPs or parameters, such as PP-YOLOE, their accuracy is not as good as ours. The proposed methods benefit from the integration of finely tuned feature capture, efficient modeling, and the reduction of unnecessary computation. Therefore, we offer well-rounded performance for drone detection.

Evaluation of Model Latency: We optimize the proposed model, and the results are in Fig. 4. The Intel i9-9900K CPU, Ubuntu 18.04.5, and NVIDIA RTX 2080 Ti GPU were used for the experiment. In the figure, the ‘Ours’ denotes our model optimized by our method. The ‘FasterR’ represents the Faster R-CNN. The ‘CascadeR’ refers to the Cascade R-CNN. The ‘PPYOLOE’ stands for the PP-YOLOE model.

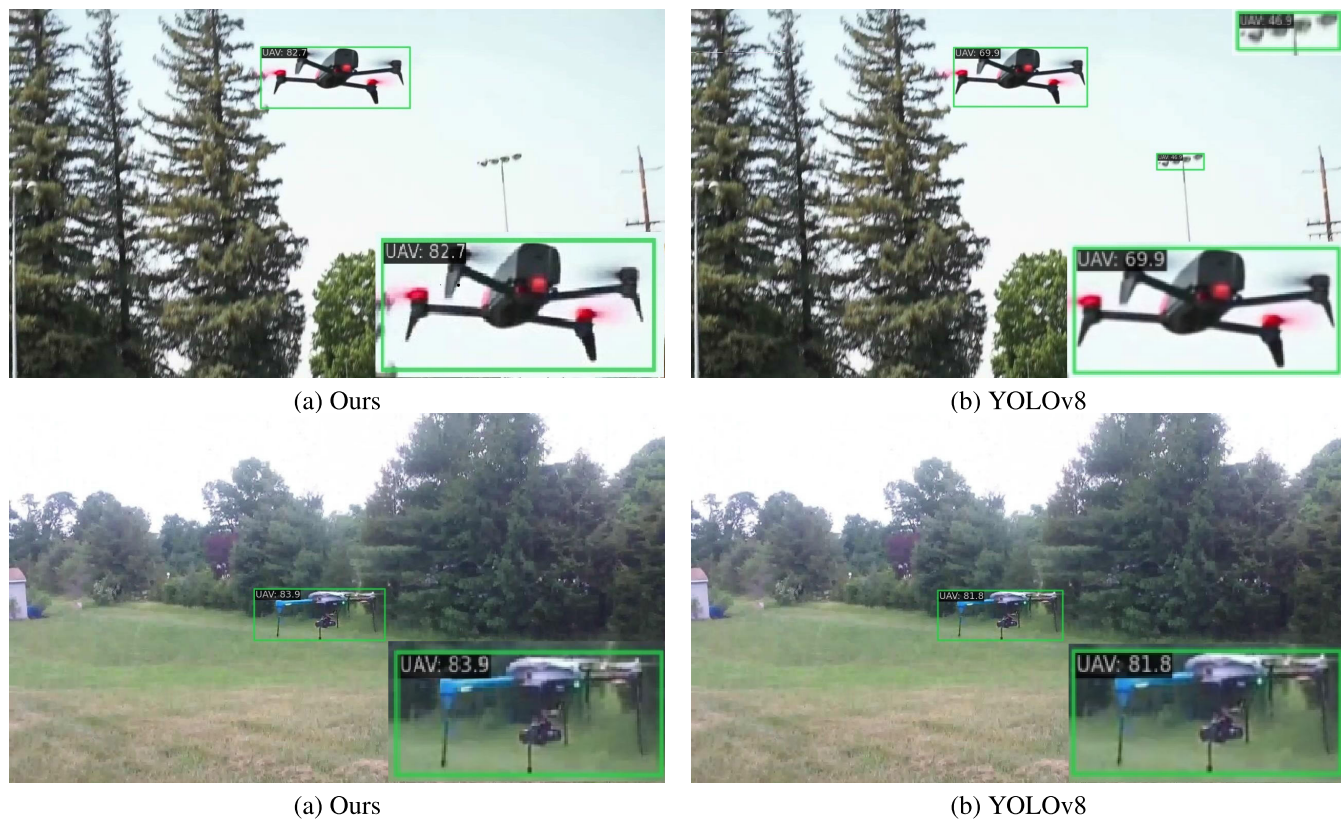


FIGURE 6. Visualization of final detection results of different models on the DUT-Anti-UAV dataset.

The ‘TVM’ denotes our model optimized by the work [33]. The latency refers to the time a model spends on drone detection, measured in milliseconds (ms). A lower latency is desirable, as it indicates faster processing and quicker results. The data suggests a significant variance in the latency of different methods. Our optimized model has a latency of only 7.20 ms, surpassing the fast YOLOX method, suggesting that our method performs exceedingly well in the speed of drone detection.

Other models, such as SSD300, PPYOLOE, and CenterNet, show relatively impressive speeds, registering latency of 9.9 ms, 12.8 ms, and 12.5 ms, respectively. SSD512 has a slightly higher latency than SSD300. Although faster than some other approaches, models such as FCOS (42.1 ms) and FPG (38.7 ms) have latency in the range of 30-50 ms, which is nowhere near as fast as our model. RetinaNet, with a latency of 43.7 ms, trails closely behind the FCOS and FPG. Faster R-CNN and DCNv2 have a latency of 44.7 and 49.4 ms, respectively. Cascade R-CNN has a notable latency of 55.0, which suggests that its speed is not its primary strength. GN + WS is the least efficient in terms of latency, with the highest value in the figure at 65.7 ms. This indicates that this method lacks the lightwightness that other models do. The models with the highest latency, like Cascade R-CNN and GN + WS, might not be suited for scenarios where processing time is critical. In contrast, our method is

an ideal choice for real-time or near-real-time applications. Additionally, compared with other optimization methods such as TVM, our optimization method speeds up 11.11%. The reason is that we performed optimized scheduling that reduces unnecessary computation. Importantly, our speedup does not affect accuracy.

### C. VISUALIZED RESULTS

In the top row of Fig. 5, we compare feature maps by Algorithm 1 and the corresponding modules of YOLOv8. In the second row, we list feature maps output by the corresponding modules of YOLOv8 and Algorithm 2, respectively. The zoomed-in sections at the bottom of each figure provide a clearer view of the object detection. For the feature maps in the top row, we show distinct hotspots (areas of high activation) around the drone, suggesting that Algorithm 1 has effectively learned to identify the key features of drones. In contrast, the corresponding module of YOLOv8 shows more widespread activations which is indicative of a less focused detection pattern. The reason is that Algorithm 1 uses an attention mechanism to focus on details and global feature capture. For the second row, our feature maps are less cluttered compared to those of the corresponding module from YOLOv8. This indicates that Algorithm 2 is more effective at filtering out irrelevant background noise and focusing on relevant features, which enhances

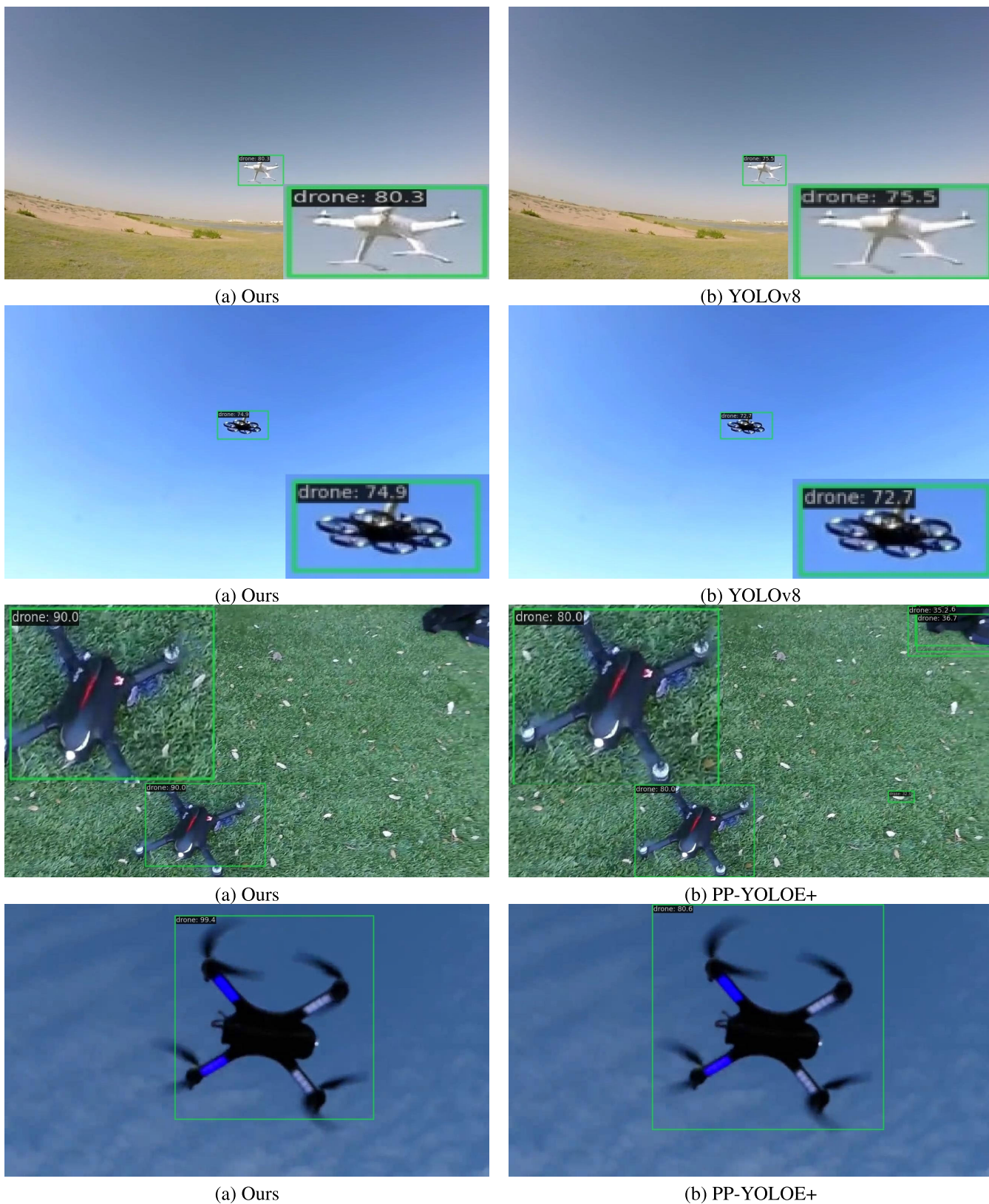


FIGURE 7. Visualization of final detection results of different models on the online drone dataset.

the model ability to detect drones with higher precision. Additionally, our boxes around the drone have a percentage score (86.9), suggesting better accuracy in drone detection than that of YOLOv8 (67.74). These comparisons suggest that Algorithm 1 and 2 are more suitable for scenarios where high precision are required in drone detection, particularly in environments where background elements might otherwise interfere with object recognition.

The final visual detection results of different models were compared on DUT-Anti-UAV [30] and Online Drone [31] datasets. Due to limited space, our model and similar models [12], [15] were selected for comparison, as shown in Fig. 6 and 7. The box in the lower right corner of the figures is the enlarged detection result. The position and predicted probability of the proposed model are more accurate than the methods [12], [15]. For the DUT-Anti-UAV test dataset, we used different images for visualization. The results are shown in Fig. 6. Our detection frame position and recognition accuracy are more precise than that of others. Our method depicts the boxes that compactly surround drones. The boxes appear to match the real object location. The length and width of pixels occupied by drones are relatively balanced in our visualization. The target size recognized is moderate. On the test set of Online Drone, we also adopted different images and methods for comparison. On Online Drone, despite the varying sizes of drones, our model achieves high accuracy, because it effectively captures and models both the intricate details and broader relationships within the feature set, ensuring comprehensive understanding and discrimination of drone features, as shown in Fig. 7.

## VI. CONCLUSION

This paper solves these drone detection challenges, including effective feature extraction and modeling, model deployment, and accelerated inference. Specifically, to acquire regional and overarching features, we introduce a FCM that encompasses local and global attention branches with a few FLOPs and parameters to detect drones. Then, we propose the FMM without extra parameters to calculate the attention weights, aiding in mitigating the influence of anomalies and extreme values. The two proposed modules effectively capture and model features. The ablation study confirms that the average recall and precision are both improved by using our modules. The evaluation on two benchmarks shows that our methods are competitive in accuracy, parameter quantities, FLOPs, and visual effects. We also employ a neuron network compiler for computational abstraction and shift scheduling decisions from runtime to compilation. The compiler maximized hardware utilization, resulting in reduced inference time for drone detection. With the compilation optimization, our method achieves faster drone detection than before. By solving these emerging challenges in drone technology, this paper contributes to safeguarding against malicious drone activities. This research has potential practical applications in enhancing security measures in various fields. For example, the proposed methods can protect critical infrastructure,

monitor borders, and ensure public safety. Future work will focus on further improving our robustness and adaptability.

## REFERENCES

- [1] X.-F. Zhu, T. Xu, J. Zhao, J.-W. Liu, K. Wang, G. Wang, J. Li, Q. Wang, L. Jin, Z. Zhu, J. Xing, and X.-J. Wu, "Evidential detection and tracking collaboration: New problem, benchmark and algorithm for robust anti-UAV system," 2023, *arXiv:2306.15767*.
- [2] B. K. S. Isaac-Medina, M. Poysier, D. Organisciak, C. G. Willcocks, T. P. Breckon, and H. P. H. Shum, "Unmanned aerial vehicle visual detection and tracking using deep neural networks: A performance benchmark," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. Workshops (ICCVW)*, Oct. 2021, pp. 1223–1232, doi: [10.1109/ICCVW54120.2021.00142](https://doi.org/10.1109/ICCVW54120.2021.00142).
- [3] J. Zhao, G. Wang, J. Li, L. Jin, N. Fan, M. Wang, X. Wang, T. Yong, Y. Deng, Y. Guo, S. Ge, and G. Guo, "The 2nd anti-UAV workshop & challenge: Methods and results," 2021, *arXiv:2108.09909*.
- [4] G. Lykou, D. Moustakas, and D. Gritzalis, "Defending airports from UAS: A survey on cyber-attacks and counter-drone sensing technologies," *Sensors*, vol. 20, no. 12, p. 3537, Jun. 2020, doi: [10.3390/s20123537](https://doi.org/10.3390/s20123537).
- [5] X. Yan, T. Fu, H. Lin, F. Xuan, Y. Huang, Y. Cao, H. Hu, and P. Liu, "UAV detection and tracking in urban environments using passive sensors: A survey," *Appl. Sci.*, vol. 13, no. 20, p. 11320, Oct. 2023.
- [6] Y. Dai, F. Giesecke, S. Oehmcke, Y. Wu, and K. Barnard, "Attentional feature fusion," in *Proc. IEEE Winter Conf. Appl. Comput. Vis. (WACV)*, Jan. 2021, pp. 3559–3568.
- [7] M. Misbah, M. U. Khan, Z. Yang, and Z. Kaleem, "TF-Net: Deep learning empowered tiny feature network for night-time UAV detection," in *Proc. Int. Conf. Wireless Satell. Syst. Cham, Switzerland: Springer*, 2023, pp. 3–18, doi: [10.1007/978-3-031-34851-8\\_1](https://doi.org/10.1007/978-3-031-34851-8_1).
- [8] H. Wang, Y. Xu, Y. He, Y. Cai, L. Chen, Y. Li, M. A. Sotelo, and Z. Li, "YOLOv5-fog: A multiobjective visual detection algorithm for fog driving scenes based on improved YOLOv5," *IEEE Trans. Instrum. Meas.*, vol. 71, pp. 1–12, 2022, doi: [10.1109/TIM.2022.3196954](https://doi.org/10.1109/TIM.2022.3196954).
- [9] M. U. Khan, M. Dil, M. Misbah, F. A. Orakazi, M. Z. Alam, and Z. Kaleem, "TransLearn-YOLOx: Improved-YOLO with transfer learning for fast and accurate multiclass UAV detection," in *Proc. Int. Conf. Commun., Comput. Digit. Syst. (C-CODE)*, May 2023, pp. 1–7, doi: [10.1109/c-code58145.2023.10139896](https://doi.org/10.1109/c-code58145.2023.10139896).
- [10] C.-Y. Wang, A. Bochkovskiy, and H.-Y.-M. Liao, "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)*, Jun. 2023, pp. 7464–7475, doi: [10.1109/CVPR52729.2023.00721](https://doi.org/10.1109/CVPR52729.2023.00721).
- [11] Y. Li, Q. Fan, H. Huang, Z. Han, and Q. Gu, "A modified YOLOv8 detection network for UAV aerial image recognition," *Drones*, vol. 7, no. 5, p. 304, May 2023, doi: [10.3390/drones7050304](https://doi.org/10.3390/drones7050304).
- [12] Ultralytics. (2023). *YOLOv8*. [Online]. Available: <https://github.com/ultralytics/ultralytics>
- [13] Z. Ge, S. Liu, F. Wang, Z. Li, and J. Sun, "YOLOx: Exceeding YOLO series in 2021," 2021, *arXiv:2107.08430*.
- [14] S. Xu, X. Wang, W. Lv, Q. Chang, C. Cui, K. Deng, G. Wang, Q. Dang, S. Wei, Y. Du, and B. Lai, "PP-YOLOE: An evolved version of YOLO," 2022, *arXiv:2203.16250*.
- [15] C. Yao, X. Liu, J. Wang, and Y. Cheng, "Optimized design of EdgeBoard intelligent vehicle based on PP-YOLOE+," *Sensors*, vol. 24, no. 10, p. 3180, May 2024.
- [16] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "SSD: Single shot MultiBox detector," in *Proc. 14th Eur. Conf. Comput. Vis. (ECCV)*, Amsterdam, The Netherlands, Cham, Switzerland: Springer, Oct. 2016, pp. 21–37, doi: [10.1007/978-3-319-46448-0\\_2](https://doi.org/10.1007/978-3-319-46448-0_2).
- [17] Z. Tian, C. Shen, H. Chen, and T. He, "FCOS: Fully convolutional one-stage object detection," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2019, pp. 9626–9635, doi: [10.1109/ICCV.2019.00972](https://doi.org/10.1109/ICCV.2019.00972).
- [18] X. Zhou, D. Wang, and P. Krähenbühl, "Objects as points," 2019, *arXiv:1904.07850*.
- [19] X. Zhu, H. Hu, S. Lin, and J. Dai, "Deformable ConvNets v2: More deformable, better results," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 9300–9308, doi: [10.1109/CVPR.2019.00953](https://doi.org/10.1109/CVPR.2019.00953).
- [20] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, "Focal loss for dense object detection," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 42, no. 2, pp. 318–327, Feb. 2020, doi: [10.1109/TPAMI.2018.2858826](https://doi.org/10.1109/TPAMI.2018.2858826).

- [21] L. Yang, R.-Y. Zhang, L. Li, and X. Xie, "SimAM: A simple, parameter-free attention module for convolutional neural networks," in *Proc. ICML Deep Learn. (ICML)*, vol. 139, 2021, pp. 11863–11874.
- [22] L. Ma, Z. Xie, Z. Yang, J. Xue, Y. Miao, W. Cui, W. Hu, F. Yang, L. Zhang, and L. Zhou, "Rammer: Enabling holistic deep learning compiler optimizations with rTasks," in *Proc. 14th USENIX Symp. Operating Syst. Design Implement. (OSDI)*. USENIX Association, 2020, pp. 881–897. [Online]. Available: <https://www.usenix.org/conference/osdi20/presentation/ma>
- [23] S. Ren, K. He, R. Girshick, and J. Sun, "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](https://doi.org/10.1109/TPAMI.2016.2577031).
- [24] Z. Cai and N. Vasconcelos, "Cascade R-CNN: Delving into high quality object detection," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 6154–6162, doi: [10.1109/CVPR.2018.00644](https://doi.org/10.1109/CVPR.2018.00644).
- [25] K. Chen, Y. Cao, C. Change Loy, D. Lin, and C. Feichtenhofer, "Feature pyramid grids," 2020, *arXiv:2004.03580*.
- [26] S. Qiao, H. Wang, C. Liu, W. Shen, and A. Yuille, "Micro-batch training with batch-channel normalization and weight standardization," 2019, *arXiv:1903.10520*.
- [27] U. Seidaliyeva, D. Akhmetov, L. Iipbayeva, and E. T. Matson, "Real-time and accurate drone detection in a video with a static background," *Sensors*, vol. 20, no. 14, p. 3856, Jul. 2020, doi: [10.3390/s20143856](https://doi.org/10.3390/s20143856).
- [28] P. Dong, C. Wang, Z. Lu, K. Zhang, W. Wan, and J. Sun, "S-feature pyramid network and attention model for drone detection," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, Jun. 2023, pp. 1–2, doi: [10.1109/ICASSP49357.2023.10096226](https://doi.org/10.1109/ICASSP49357.2023.10096226).
- [29] H. Wang, X. Wang, C. Zhou, W. Meng, and Z. Shi, "Low in resolution, high in precision: UAV detection with super-resolution and motion information extraction," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, Jun. 2023, pp. 1–5, doi: [10.1109/ICASSP49357.2023.10096656](https://doi.org/10.1109/ICASSP49357.2023.10096656).
- [30] J. Zhao, J. Zhang, D. Li, and D. Wang, "Vision-based anti-UAV detection and tracking," *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 12, pp. 25323–25334, Dec. 2022, doi: [10.1109/TITS.2022.3177627](https://doi.org/10.1109/TITS.2022.3177627).
- [31] Y. Chen, P. Aggarwal, J. Choi, and C.-C. J. Kuo, "A deep learning approach to drone monitoring," in *Proc. Asia-Pacific Signal Inf. Process. Assoc. Annu. Summit Conf. (APSIPA ASC)*, Dec. 2017, pp. 686–691. [Online]. Available: <https://github.com/chelicynly>
- [32] M. Contributors. (2022). *MMYOLO: OpenMMLab YOLO Series Toolbox and Benchmark*. [Online]. Available: <https://github.com/openmmlab/mmyolo>
- [33] T. Chen, L. Zheng, E. Yan, Z. Jiang, T. Moreau, L. Ceze, C. Guestrin, and A. Krishnamurthy, "Learning to optimize tensor programs," in *Proc. Adv. Neural Inf. Process. Syst. (NIPS)*, 2018, pp. 3393–3404.



**XIAOHAN TU** (Member, IEEE) received the M.Sc. and Ph.D. degrees in computer science and technology from Hunan University, Changsha, China, in 2017 and 2021, respectively. She is currently a Lecturer with Zhengzhou Police University. She has participated in the project of the National Natural Science Foundation of China: "CPS Instantiation-Research on the Smart Inspection Robot of Catenary." She has authored or co-authored more than ten research papers at IEEE international conferences or journals. Her current research interests include cyber-physical systems, computer vision, and machine learning. She is currently a reviewer of many journals.



**CHUANHAO ZHANG** received the Ph.D. degree in electronic science and technology from the Huazhong University of Science and Technology, China, in 2009. He is currently an Associate Professor with Zhengzhou Police University. He won the title of Academic Technology Leader of Henan Province, in 2020. His research interests include the application of big data and artificial intelligence technology in police.



**HAIYAN ZHUANG** received the M.Sc. degree from Zhengzhou University, China, in 2006. She is currently an Associate Professor with Zhengzhou Police University, China. She has achieved some work in developing systems and methodologies that leverage large datasets to enhance the efficiency and effectiveness of police work. Her research interests include computer vision, big data, cyber security, and artificial intelligence.



**SIPING LIU** received the M.Sc. and Ph.D. degrees in computer science and technology from Hunan University, China, in 2017 and 2022, respectively. He is currently an Assistant Researcher with Zhengzhou University, China. He has not only been at the forefront of imparting knowledge but also actively involved in advancing research in the field of electronic science. His pedagogical approach combines rigorous academic theory with real-world applicability, ensuring that his students are well-equipped to meet the challenges of modern CPS. His research interests include embedded and cyber-physical systems (CPS), cybersecurity, and machine learning.



**RENFA LI** (Senior Member, IEEE) is currently a Professor of computer science and electronic engineering with Hunan University, Changsha, China. He is also the Director of the Key Laboratory for Embedded and Network Computing of Hunan Province, China. His current research interests include computer architectures, embedded computing systems, cyber-physical systems, and the Internet of Things.

Dr. Li is a member of the Council of China Computer Federation and a Senior Member of the Information Processing Society of Japan.

• • •