DRNet: A Miniature and Resource-Efficient MAV Detector

Xunkuai Zhou, *Member, IEEE*, Bingxin Han, Li Li, *Member, IEEE*, Jie Chen, *Fellow, IEEE*, and Ben M. Chen, *Fellow, IEEE*

***Abstract*— This article focuses on the challenge of accurate microaerial vehicle (MAV) detection under memory-constrained and lower computational cost conditions. A miniature MAV detection framework, DRNet, is proposed to address this issue. DRNet incorporates a dual skip concatenation (DSC) network and squeezing excitation residual (SER) networks for feature extraction, which enhances the model's representation capacity. Additionally, spatial attention computation improves object localization and representation capabilities. A lightweight network facilitates efficient feature fusion, further optimizing DRNet's performance. DRNet's superior performance over other methods is validated across four challenging datasets. DRNet achieves a comparable accuracy-matching heavyweight method while saving 99.97% parameters, and a compact model size of just 309 kB makes it the smallest high-accuracy, low-computational-requirement MAV detector to date. Furthermore, DRNet can reduce computational costs by 95.3% and GPU memory usage by 50% when processing high-resolution images with dimensions of 1280 × 1280. Challenging real-world tests and experimental deployments on edge-computing devices further confirm DRNet's feasibility and portability.**

***Index Terms*— Edge computing, energy-efficient, microaerial vehicle (MAV), miniature deep learning.**

I. Introduction

VISION-BASED microaerial vehicle (MAV) detection and confidence measurement are crucial in various applications, such as flight safety, collision avoidance, autonomous navigation in air transportation, multi-MAV visual formation, and privacy protection [1]. The ability to detect small targets such as MAVs is a critical performance metric for assessing detection methods [2]. Existing advanced detectors require significant computational resources and memory footprint, challenging their implementation on memory-constrained edge-computing devices, while low-power consumption detectors typically suffer from insufficient accuracy. A compact MAV detection framework that ensures efficient and accurate detection with reduced computational overhead and memory usage can facilitate its application in edge-oriented microdeep learning systems.

Several innovative detectors reduce the computational cost by downscaling the input size by a large ratio. For example, TinyDet employs a 320 × 320 input size and a 5 × 5 feature map [6] for detection. ThunderNet [7] uses a 320 × 320 input but relies solely on a 20 × 20 feature map for detection. Pelee [8] processes a 304 × 304 input, with the largest feature map for detection being 19 × 19. The smallest variant within TinyDet, TinyDet-S, still has a larger model size of 19.8 MB. While employing smaller input images reduces computational overhead, it results in less detailed feature maps, thereby diminishing the effectiveness of small object detection. Conversely, larger detectors like Faster R-CNN [9] utilize an input size of 800 × 1333, yielding a larger feature map of 200 × 333 to enhance detection accuracy for small targets. However, the substantial model size of Faster R-CNN imposes a burden on memory-constrained devices and increases computational complexity. In summary, while smaller input sizes enhance efficiency, they diminish detection accuracy; larger models, though more accurate, increase computational costs due to higher complexity [10]. Therefore, *exploring a framework that balances the accuracy of large models with the low complexity and low memory usage of small models will enhance the deployment and application of MAV detection methods on memory-constrained devices*.

This work proposes a network for MAV detection, namely, DRNet, an end-to-end detector optimized for memory-efficient and accurate detection of MAVs with high-resolution feature maps. We utilize squeezing excitation residual (SER) networks followed by a dual skip concatenation (DSC) network for feature extraction and incorporate a feature fusion network for enhancing the feature map. The dropout operation in each SER can mitigate the risk of overfitting, and channel attention computation can enhance the feature representation ability [11]. The DSC network design can enhance small object detection performance; however, it may also increase computational overhead. To balance computational overhead and efficacy, only one DSC network is integrated after the SER networks in DRNet. A max-pooling layer connects the SER to the DSC, acting as a downsampling mechanism. A lightweight feature fusion network is designed to enhance multiscale object detection. Spatial attention computation is applied before the fusion operation to improve MAV

Received 21 October 2024; revised 3 December 2024; accepted 19 December 2024. Date of publication 5 March 2025; date of current version 19 March 2025. This work was supported in part by the Research Grants Council of Hong Kong, SAR, under Grant 14217922. The Associate Editor coordinating the review process was Dr. Sudao He. *(Corresponding author: Li Li.)*

Xunkuai Zhou is with the School of Electronics and Information Engineering, Tongji University, Shanghai 201804, China, and also with the Department of Mechanical and Automation Engineering, The Chinese University of Hong Kong, Hong Kong, China (e-mail: 2010474@tongji.edu.cn).

Bingxin Han and Ben M. Chen are with the Department of Mechanical and Automation Engineering, The Chinese University of Hong Kong, Hong Kong, China (e-mail: 1155186689@link.cuhk.edu.hk; bmchen@cuhk.edu.hk).

Li Li and Jie Chen are with the School of Electronics and Information Engineering, Tongji University, Shanghai 201804, China (e-mail: lili@tongji.edu.cn; chenjie206@tongji.edu.cn).

Digital Object Identifier 10.1109/TIM.2025.3548216

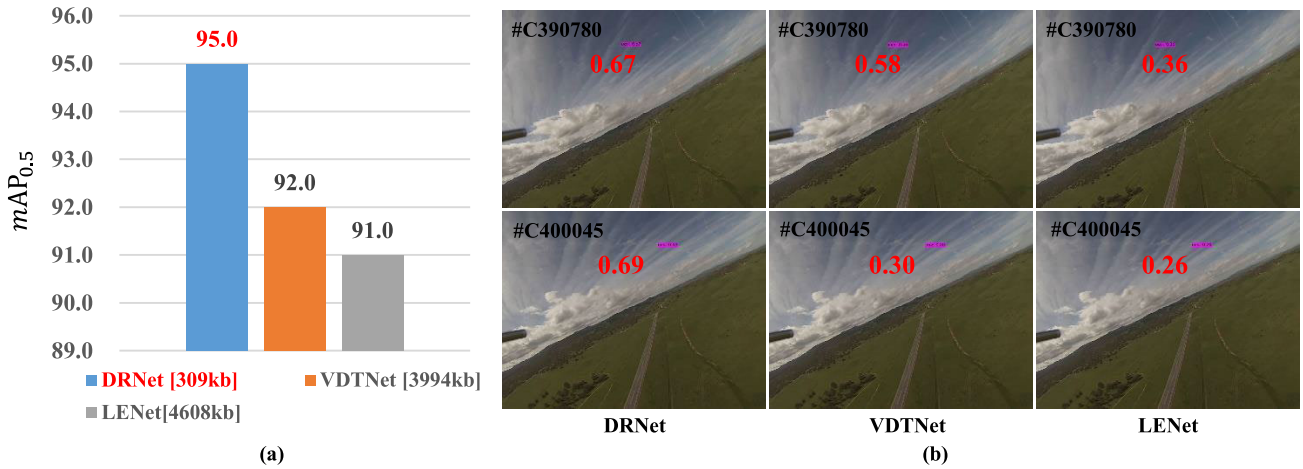


Fig. 1. (a) Model size versus accuracy on NPS-Drone [3]. The numerical values within the brackets indicate the model size, while the numbers above the bars represent the accuracy. Our DRNet achieves higher mAP with significantly fewer parameters compared with other detectors [4], [5]. (b) Qualitative real-world detection visualization results. DRNet can detect MAVs with higher confidence than VDTNet and LENet. Please zoom in for the best view.

localization and representation ability by filtering redundant information [12]. The carefully designed feature extraction and fusion networks enjoy fewer channels to reduce parameters and computational costs. The detection head employs five anchors per grid cell, covering a broader detection area and enhancing overall accuracy. Performance evaluations on four challenging datasets confirm the proposed method's detection capabilities.

As shown in Fig. 1(a), the proposed DRNet's accuracy is compared with advanced methods such as VDTNet [4] and LENet [5]. The number in square brackets indicates the model size, with accuracy numbers displayed above the bars. DRNet surpasses VDTNet's accuracy by 3% with a significantly smaller model size (309 kB versus 3994 kB), just one-tenth of its size. Similarly, DRNet exceeds LENet's accuracy by 4%, with less than one-tenth of the model size (309 kB versus 4608 kB), making it highly suitable for deployment on memory-constrained devices.

Fig. 1(b) shows effective MAV detection visualizations, where each row represents the same frame from a video, and each column represents detection results from DRNet, VDTNet, and LENet, respectively. The first row displays frame 780 of video 39, and the second row displays frame 45 of video 40. Red numbers indicate target detection confidence. DRNet detects targets with higher confidence than VDTNet and LENet, establishing it as the smallest and most accurate MAV detection method with superior practical performance.

In summary, the contributions of this work are as follows.

1) We propose DRNet for accurate and efficient MAV detection on high-resolution feature maps. DRNet achieves modeling capability comparable to the advanced TransVisDrone [13] without many parameters, while maintaining lower memory usage and latency. Specifically, DRNet saves 99.97% parameters compared to TransVisDrone.
2) We design an SER network and a DSC network for enhancing feature extraction and representation ability, utilizing spatial attention calculation to filter redundant information and then design a lightweight feature fusion for enhancing the multiscale object detection.
3) Detailed ablation experiments on multiple challenging datasets validate the effectiveness of the SER and DSC networks and the rationale of DRNet. Qualitative visualizations further confirm that the designed strategies achieve the expected results.
4) Extensive comparisons on four challenging datasets show that the method balances accuracy and speed with reduced computational cost, achieving state-of-the-art performance on high-resolution images. Evaluations across various scenarios, including small MAV detection, low-light, and camouflage, confirm its practical effectiveness.
5) Deployment experiments on edge devices demonstrate portability and feasibility for practical applications. Comparative results show inference speeds of 14.4 frames/s on GPU and 2.7 frames/s on CPU, indicating efficiency even on CPUs with the attention operation. The inference speed on the CPU is ten times faster than that of advanced methods.

The remainder of this article is structured as follows. Section II discusses related works. Section III provides a detailed introduction to the model. Section IV presents the experimental results and evaluates the performance. Finally, Section V provides the conclusion and describes future work.

II. Related Works

A. Lightweight Object Detector

In recent years, publicly accessible lightweight models for object detection have developed rapidly. Advanced lightweight design methodologies have propelled the evolution of these approaches [14]. Lightweight detection networks frequently serve as feature extractors. For example, integrating MobileNet [15] into Faster R-CNN [9] reduces the model size by 171 MB compared to using ResNet50 [16] as the feature extraction network. Similarly, applying MobileNet to Cascade R-CNN [17] reduces the model size by

approximately 200 MB. However, this reduction in model size can compromise accuracy, which may not justify the tradeoff for accurate object detection. Many lightweight detectors adopt specialized backbones tailored to the specific demands of detection tasks, inspired by established classification networks. Additionally, well-structured object detection pipelines provide a solid foundation for developing lightweight detection mechanisms. Predominantly, these lightweight detectors [8], [18], [19] adopt a streamlined one-stage architecture. For example, PeleeNet [8] exclusively employs traditional convolutions, foregoing the popular mobile convolutions. YOLObile [19] is specifically designed for mobile real-time processing in resource-constrained environments, featuring optimizations for such scenarios. On the other hand, while two-stage detectors are generally more complex and slower, some research indicates they can rival the efficiency of one-stage models when the second stage is designed to be lightweight [6], [20]. Although this two-stage method excels in detecting small objects due to its intricate design, its extensive parameters and complex strategies often lead to higher computational complexity and memory footprint, which poses challenges for deployment on memory-limited edge devices.

B. MAV Detection

Traditional detection methods such as radar and radio frequency are limited in their ability to detect MAVs due to the weak reflection signal [21]. Computer vision technology can overcome the limitations of traditional detection methods. The Dogfight [22] method accurately detects MAVs and performs excellently across two public datasets. However, its inference speed of 1 frames/s on the advanced processor NVIDIA RTX A6000 limits practical deployment on edge-computing devices. TransVisDrone effectively detects MAVs on edge-computing devices, with its performance validated across three datasets. However, TransVisDrone's nearly 1-GB model size imposes a substantial memory burden on detection systems.

Cheng et al. [23] improve memory efficiency by adopting MobileViT as the feature extractor and PANet refinement to increase detection accuracy. Modifications to the SAG-YOLOv5s [24] reduce the input resolution of YOLOv5 to 96×96, resulting in a compact 15M model that minimizes computational demands and accelerates inference. However, SAG-YOLOv5s manages an inference speed of only 15 frames/s on an advanced processor RTX 2070 SUPER. Meanwhile, DTD-YOLOv4-Tiny [25] incorporates *ShuffleNet* for feature extractor and applies k-means clustering to fine-tune anchor configurations, thereby enhancing throughput and diminishing parameters. However, DTD-YOLOv4-Tiny falls short of the desired accuracy levels. Sun et al. [26] improve the lightweight EXTD [27] by integrating spatial attention modules, creating TIBNet, which surpasses DTD-YOLOv4-Tiny in accuracy with a smaller 697-kB model size. Nevertheless, TIBNet's substantial 290-ms inference delay poses challenges for practical deployment. Fang et al. [28] propose an infrared drone detection method based on depth-wise separable residual dense network and multiscale feature fusion, but it still faces higher computational complexity and slower inference speed.

We identify a gap in the literature concerning lightweight detectors that have low memory and computational costs yet match the performance of heavier detectors, highlighting an urgent issue for further exploration. This article aims to achieve accurate MAV detection with limited computational resources and a reduced memory footprint.

III. Methodology

Motivation: This study aims to resolve the tradeoff between low parameters and high accuracy by designing a method that can be accurately deployed on devices with extreme memory constraints. It addresses high memory and computational demands when processing high-resolution inputs. The method achieves high accuracy and edge deployability for MAV detection, matching the accuracy of larger models while reducing parameters and computational overhead. The specific design methodology will be detailed in Sections III-A–III-C.

A. DRNet Framework

As illustrated in Fig. 2, the DRNet framework is composed primarily of the SER module, convolutional layers with diverse activation functions, Dropout layers, max-pooling layers, upsampling layers, a DSC network, and two detection branches. One detection branch employs a spatial attention mechanism to filter redundant information. The notation $2\times$ signifies that two modules are cascaded sequentially. The SER module augments the model's perceptual capabilities by adaptively assigning weights to feature channels, thereby enhancing feature representation across varying scales. In particular, the SER module aggregates global information from the input feature map through global average pooling to generate a compact global feature vector. This vector subsequently passes through two fully connected layers and, utilizing a Sigmoid activation function, produces weights for each feature channel. These weights are then applied to the original feature map, enabling dynamic adjustment of feature responses across channels as needed, thus reinforcing the model's representation power.

In DRNet, convolutional layers are employed to progressively extract input features, forming a hierarchical structure of abstract semantic representations. The activation functions associated with these convolutional layers include LeakyReLU, Linear, Mish [29], and Sigmoid. For the detection of small objects—where target information is limited and redundant information is plentiful—an initial extraction of more features is necessary, followed by suppression and filtering of redundant information in a coarse-to-fine manner. Consequently, early convolutional layers predominantly utilize the LeakyReLU activation function, which retains a gradient in the negative direction. While the Mish function also retains a gradient in the negative direction, it simultaneously restricts some gradient propagation, thus filtering out portions of the redundant information. By stacking these convolutional layers, a deep network architecture is constructed, enabling the model to capture intricate spatial information and structural details.

To further enhance the network's discriminative power, DRNet incorporates a DSC network, as depicted in the blue

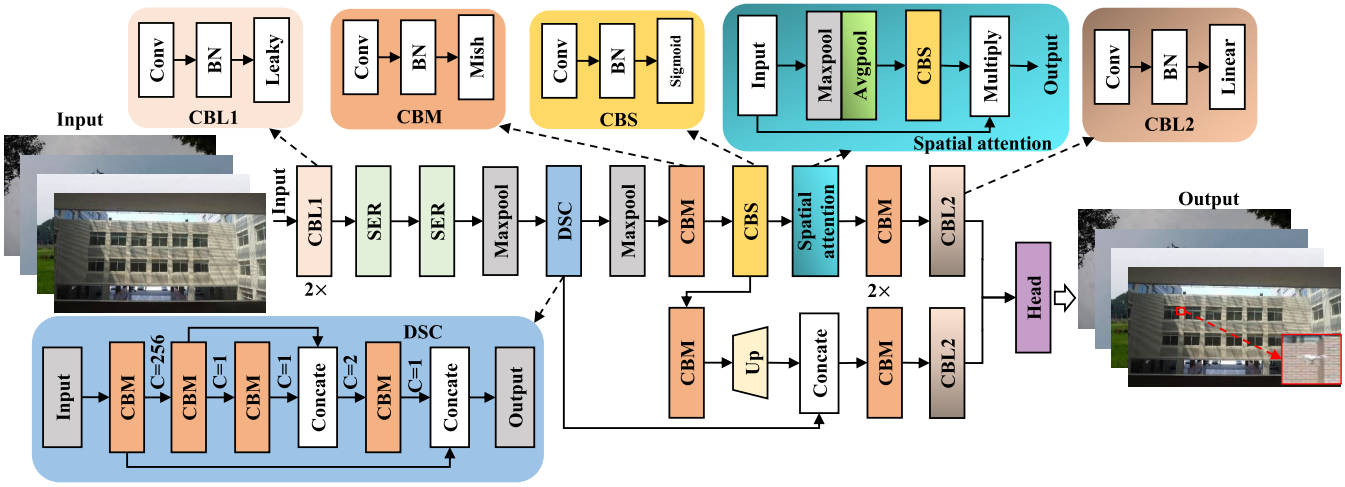


Fig. 2. DRNet framework. The framework comprises a total of 47 layers, primarily including the SER module, convolutional layers with various activation functions, dropout layers, a max-pooling layer, an upsampling layer, a DSC module, and two detection heads.

block at the bottom left of Fig. 2. This module fuses features from two different convolutional layers, thereby preserving finer details. The channel count in each DSC layer is generally limited to 1 or 2, with a maximum of 256, optimizing memory usage and reducing the model's overall footprint.

In the final stage of the framework, two detection branches are implemented to extract features at different scales, with each branch tailored to handle features of a specific resolution, thereby enhancing the accuracy of multiscale detection. Notably, the first branch incorporates a spatial attention calculation, which is essential for filtering redundant information and improving localization quality. By selectively focusing on the most relevant regions within the feature map, the spatial attention calculation effectively suppresses irrelevant noise and enhances the representation of key features, ultimately contributing to improved detection accuracy. This approach enables the framework to capture significant patterns across various scales while preserving overall detection performance. The channels in each CBL2 layer (a combination of convolutional, layer normalization, and linear activation function) adjacent to the detection head are represented as $C = (N + 5) \times 5$, where N denotes the number of categories to be detected. Each grid generates five prior boxes, compared to the typical detection method that produces only three prior boxes. By generating five prior boxes, the model achieves greater coverage of target ranges, thereby enhancing detection accuracy.

Table I details the specific network layers and parameters of DRNet. DRNet comprises a total of 47 layers, where "CBX" denotes a convolutional layer with "X" representing the activation function. The remaining layers are explained as follows.

1) *C:* Channels.
2) *K/S:3/1:* Kernel or pooling size is 3×3, and the stride is 1.
3) *Route:* Single parameter i indicates that a network branch is derived from the ith layer. Multiple parameters $i, \ldots, j$ indices the features of the ith to $j - 0$ layers are concatenated.

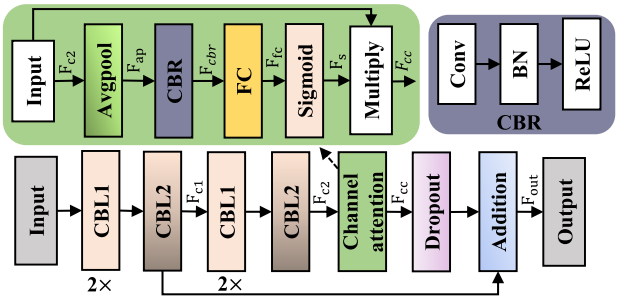


Fig. 3. SER module.

4) *Shortcut:* Single parameter i, indicates that the current layer is elementwise added with the ith layer.
5) *Scale:* Single parameter i, indicates that the current layer is connected with the the ith layer.
6) *Sam:* Spatial attention computation.
7) *Upsample:* Increase the spatial dimension of the input data with a given factor using interpolation.
8) *Head:* Output the detected objects' confidence scores, labels, and bounding boxes.

B. SER Module

As depicted in Fig. 3, each SER module comprises six convolutional layers and a Dropout layer, with channel attention computations interconnecting the convolutional and Dropout layers. ReLU serves as the activation function. The Dropout layer employs a neuron dropout probability of 0.15 to mitigate overfitting, thereby enhancing the model's generalization capability by randomly deactivating certain neurons during training. Notably, within the SER module, a linear activation function follows the addition operation, bolstering the model's robustness in feature processing. The feature extraction process of SER can be formulated as follows:

$$\begin{aligned} F_{c1} &= \mathrm{CBL}_2(\mathrm{CBL}_1(\mathrm{CBL}_1(F_{\mathrm{in}}))) \\ F_{c2} &= \mathrm{CBL}_2(\mathrm{CBL}_1(\mathrm{CBL}_1(F_{c1}))) \\ F_{cc} &= \mathrm{CA}(F_{c2}) \\ F_{\mathrm{out}} &= \mathrm{Dropout}(F_{cc}) + F_{c1} \end{aligned} \tag{1}$$

TABLE I
NUMBER OF CHANNELS AND FILTERS OF THE DRNET

0:CBLeaky		**1:CBLeaky**		**2:CBLeaky**		**3:CBLinear**		**4:CBLeaky**		**5:CBLeaky**	
C:8	K/S:3/2	C:8	K/S:1/1	C:8	K/S:3/1	C:4	K/S:1/1	C:8	K/S:1/1	C:8	K/S:3/1
6:CBLinear		**7:avg**		**8:CBRelu**		**9:CB**		**10:Scale**		**11:dropout**	
C:4	K/S:1/1	-	-	C:8	K/S:1/1	C:4	K/S:1/1	C:32-	6	-	0.15
12:Shortcut		**13:CBLeaky**		**14:CBLeaky**		**15:CBLinear**		**16:CBLeaky**		**17:CBLeaky**	
-	-	C:24	K/S:1/1	C:24	K/S:3/2	C:8	K/S:1/1	C:32	K/S:1/1	C:32	K/S:3/1
18:CBLinear		**19:avg**		**20:CBRelu**		**21:CB**		**22:Scale**		**23:dropout**	
C:8	K/S:1/1	-	-	C:8	K/S:1/1	C:8	K/S:1/1	-	18	-	0.15
24:Shortcut		**25:maxpool**		**26:CBMish**		**27:Route**		**28:CBMish**		**29:CBMish**	
-	-	-	K/S:2/2	C:256	K/S:3/1	-	26	C:1	K/S:3/1	C:1	K/S:3/1
30:Route		**31:CBMish**		**32:Route**		**33:maxpool**		**34:CBMish**		**35:CB**	
29	28	C:1	K/S:1/1	26	31	-	K/S:2/2	C:9	K/S:3/1	C:9	K/S:7/1
36:Sam		**37:CBMish**		**38:CBMish**		**39:CBLinear**		**40:Head**		**41:Route**	
-	34	C:9	K/S:1/1	C:10	K/S:3/1	C:30	K/S:1/1	-	-	-	35
42:CBMish		**43:Upsample**		**44:Route**		**45:CBMish**		**46:CBLinear**		**47:Head**	
C:128	K/S:1/1	-	2x	43	26	C:8	K/S:3/1	C:30	K/S:1/1	-	-

in this context, $F_{\text{in}} \in \mathbb{R}^{C\times W\times H}$ represents the input feature maps, $F_{c1} \in \mathbb{R}^{C/2\times W\times H}$ denotes the output feature maps from the first three convolutional layers, $F_{c2} \in \mathbb{R}^{C/2\times W\times H}$ represents the output feature maps from the sixth convolutional layer, $F_{\text{cc}} \in \mathbb{R}^{C/2\times W\times H}$ indicates the output feature maps after applying attention calculation to F_{c2}, and $F_{\text{out}} \in \mathbb{R}^{C/2\times W\times H}$ is the final outputs of the SER.

The channel attention calculation primarily augments the performance of convolutional neural networks (CNNs) by adaptively adjusting the weights of each feature channel. This calculation involves three main steps.

1) *Global Average Pooling:* Spatial features for each channel are compressed into a global feature descriptor, reducing high-dimensional spatial features to 1-D channel features that capture global information for each channel.
2) *Adaptive Learning via Fully Connected Network:* The compressed channel features undergo adaptive learning through a fully connected network to generate weights for each channel, reflecting the relative importance of various channels.
3) *Weight Application:* These weights are subsequently applied to the original feature channels, amplifying essential feature channels while suppressing irrelevant or less important ones.

The feature extraction process of attention computation can be formulated as follows:

$$\begin{aligned} F_{\text{ap}} &= \text{GlobalAvgPool}(F_{c2}) \\ F_{\text{cbr}} &= \text{CBR}(F_{\text{ap}}) \\ A &= \sigma(\text{FC}(F_{\text{cbr}})) \\ F_{cc} &= A \odot F_{c2} \end{aligned} \tag{2}$$

where GlobalAvgPool(F_{c2}) is the global average pooling of the feature map $F_{c2} \in \mathbb{R}^{C/2\times W\times H}$. FC is a fully connected layer. σ is the sigmoid activation function to obtain attention scores in the range of [0, 1]. $\odot$ denotes the elementwise multiplication between the attention map $A \in \mathbb{R}^{C/2\times 1\times 1}$ and the feature map F_{c2}.

By implementing these steps, channel attention calculation effectively enhances the network's feature representation capacity, allowing the model to focus on channels with significant information. This approach improves the accuracy and efficiency of tasks, such as classification and detection while maintaining low computational costs.

C. DSC Module

The blue module at the bottom left of Fig. 2 represents the DSC module, which comprises four convolutional layers and two concatenation layers. This module efficiently extracts and processes input feature information through a combination of multiple convolutional layers and a feature fusion mechanism. Given the shallow network design of this study, issues, such as gradient vanishing, which are prevalent in deeper networks, are less likely to arise, and the network has a limited number of channels. To better preserve feature information, the DSC employs concatenation rather than addition. Specifically, the first feature fusion layer merges the outputs of the first and fourth convolutional layers along the channel dimension, producing a more representative intermediate feature. The second feature fusion layer then combines the outputs of the second and third convolutional layers, allowing deeper features to interact with shallower ones and thereby enhancing feature representation.

The Mish function is selected as the activation function for the convolutional layers within the DSC. As a nonlinear activation function, Mish maintains gradient flow and enhances the neural network's capacity to learn complex features. Compared to traditional activation functions (e.g., ReLU and Tanh), the Mish function significantly elevates the network's nonlinear representation ability. This enhanced nonlinearity enables the network to capture finer distinctions within the data, thereby improving overall model performance and prediction accuracy. By utilizing the Mish activation function, the DSC module retains a high level of feature representation, making it well-suited for handling complex pattern recognition tasks.

If define $F_{\text{din}} \in \mathbb{R}^{C\times W\times H}$ as the input of DSC, the feature extraction process of the DSC network is formulated as follows:

$$\begin{aligned} F_{\text{dc1}} &= \text{CBM}(F_{\text{din}}) \\ F_{\text{dc2}} &= \text{CBM}(F_{\text{dc1}}) \\ F_{\text{dc3}} &= \text{CBM}(F_{\text{dc2}}) \\ F_{\text{con1}} &= \text{Concate}[F_{\text{dc3}}, F_{\text{dc2}}] \end{aligned}$$

$$F_{\text{dc4}} = \text{CBM}(F_{\text{con1}})$$
$$F_{\text{con2}} = \text{Concate}[F_{\text{dc4}}, F_{\text{dc1}}] \tag{3}$$

where $F_{\text{dc1}} \in \text{R}^{256\times W\times H}$, $F_{\text{dc2}} \in \text{R}^{1\times W\times H}$, $F_{\text{dc3}} \in \text{R}^{1\times W\times H}$, and $F_{\text{dc4}} \in \text{R}^{1\times W\times H}$ represent the convolutional features, respectively. $F_{\text{con1}} \in \text{R}^{2\times W\times H}$ and $F_{\text{con2}} \in \text{R}^{257\times W\times H}$, respectively, represent the fusion feature from different layers.

D. Loss Function

Our framework is optimized using three distinct loss functions: 1) objectness loss, assessing the probability that a predicted bounding box contains an object; 2) classification loss, evaluating the accuracy of the predicted class for the detected object; and 3) localization loss, quantifying the precision of the bounding box coordinates relative to the true object location, as outlined in [30].

1) Objectness Loss: The feature map, denoted as $\mathbf{F}$, is partitioned into a grid consisting of $G \times G$ cells. Within each cell, S bounding boxes are predicted. The objectness loss, which evaluates the likelihood that each predicted bounding box accurately contains an object, can be mathematically formulated as follows:

$$L_{\text{obj}} = \sum_{a=0}^{G^2}\sum_{b=0}^{S} \mathbb{V}_{\text{ab}}^{\text{obj}} \left(p_a - \hat{g}_a\right)^2 + \beta_{\text{nbj}} \sum_{a=0}^{G^2}\sum_{b=0}^{S} \left(1 - \mathbb{V}_{\text{ab}}^{\text{obj}}\right)\left(p_a - \hat{g}_a\right)^2. \tag{4}$$

In the aforementioned formulation, $\mathbb{V}_{\text{ab}}^{\text{obj}}$ signifies a binary indicator function, wherein the value is set to 1 if the jth bounding box within cell a encapsulates the object. The coefficient β_{nbj} is assigned a value of 5. The terms p_a and $\hat{g}_a$ correspond to the predicted confidence scores and the ground-truth confidence scores, respectively.

2) Classification Loss: As delineated in the following equation, the class-specific loss is calculated utilizing the binary cross-entropy method:

$$L_{\text{cls}} = \sum_{i=0}^{S^2}\sum_{j=0}^{B} \mathbb{V}_{i,j}^{\text{obj}} \sum_{c\in\text{classes}} \hat{g}_i(c)\log(p_i(c)) + (1-\hat{g}_i(c))\log(1-p_i(c))). \tag{5}$$

In this context, $p_i(c)$ and $\hat{g}_i(c)$ represent the probability score associated with the predicted class and the actual class label from the ground truth, respectively.

3) Localization Loss: Similarly, the localization loss can be formulated as follows:

$$L_{\text{loc}} = 1 - \text{IOU} + \frac{\text{Dis}^2(b,\hat{b})}{c^2} + \rho k$$
$$\text{IOU} = \frac{|B\cap\hat{B}|}{|B\cup\hat{B}|}, \quad \rho = \frac{k}{(1-\text{IOU})+k}$$
$$k = \frac{4}{\pi^2}\left(\arctan\frac{\hat{\omega}}{\hat{h}} - \arctan\frac{w}{h}\right)^2. \tag{6}$$

In the specified formulation, $\hat{B} = (\hat{x}, \hat{y}, \hat{\omega}, \hat{h})$ denotes the coordinates of the ground-truth bounding box, while $B = (x, y, \omega, h)$ corresponds to the predicted bounding box. The variables b and $\hat{b}$ represent the central points of B and $\hat{B}$, respectively. The function $\text{Dis}(\cdot)$ is defined as the Euclidean distance between these central points, and c represents the diagonal length of the smallest enclosing box that covers both B and $\hat{B}$. The total loss is expressed as follows:

$$\text{Loss}_{\text{total}} = L_{\text{obj}} + L_{\text{cls}} + L_{\text{loc}}. \tag{7}$$

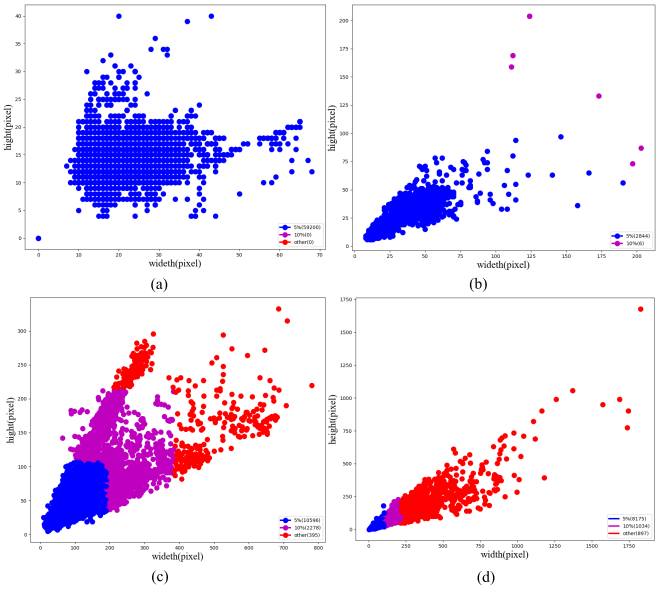


Fig. 4. This statistical data shows the sizes of target MAVs in NPS-Drone [3], TIBNet [26], Det-Fly [31], and DUT [32]. The blue points represent MAV images where the width and height are less than 5% of the total image size. The purple points indicate MAV images, where the sizes are less than 10% of the image size. The remaining red points correspond to samples, where the sizes are greater than 10% of the image size. (a) NPS-Drone. (b) TIBNet. (c) Det-Fly. (d) DUT.

IV. Experiment

A. Dataset Analysis

To enhance the practicality of the trained model, the four datasets used encompass a diverse range of scenarios. Among these, NPS-Drone [3] and Det-Fly [31] are air-to-air MAV detection datasets, TIBNet [26] is a ground-to-air MAV detection dataset, and DUT [32] contains images sourced from the Internet. The varying distances of MAVs from the camera result in a broad distribution of target sizes, closely reflecting real-world detection conditions. The distribution of target sizes across the four publicly available datasets is illustrated in Fig. 4. Blue points indicate targets with both length and width under 5% of the entire image; purple points denote targets with both dimensions under 10%; and red points represent targets whose length and width exceed 10% of the image size. Targets with a length and width smaller than 10% of the image are classified as small targets [31]. In the figure, the bottom right corner displays the quantities of blue, purple, and red points, respectively. From this statistical chart, it is apparent that NPS-Drone and TIBNet consist entirely of small targets, while Det-Fly and DUT predominantly fall within the small target range. The following is a more detailed description of the four datasets we used.

1) NPS-Drone: This dataset is released by the Naval Postgraduate School (NPS) and is publicly available. It contains 50 high-definition videos (with resolutions of 1920 × 1080 and 1280 × 760), recorded using a GoPro-3 camera mounted on a custom triangular-wing MAV. The minimum, average, and maximum sizes of the MAV are 10 × 8, 16.2 × 11.6, and 65 × 21, respectively, with the average MAV size being 0.05% of the average frame size. The dataset consists of a total of 70 250 frames/s. In the experiments, the first 40 videos are used for training and validation, while the last ten videos are used for testing.

2) TIB-Net: The dataset contains 2860 images of various types of MAVs, such as multirotor and fixed-wing MAVs. The collected images have a resolution of 1920 × 1080 pixels. The images are captured using a camera fixed on the ground, approximately 500 m away from the MAV. The scenes cover a variety of lighting conditions, including both daytime and nighttime. Additionally, each image is annotated with bounding box information in Pascal VOC format.

The dataset also includes some challenging samples, such as very small MAVs, blurred MAVs, and complex environments. Visual information alone is insufficient for accurate detection, as the size of the MAV in the dataset is much smaller than that of other common objects. Most MAV occupy less than 0.1% of the image area. In the experiments, following the original article's partitioning rule [26], 75% of the collected data are selected as the training set, with the remaining portion used as the test set.

3) Det-Fly: Det-Fly consists of 13 271 images of target MAVs (DJI Mavic). Each image has a resolution of 3840 × 2160 pixels. Some of the images in the dataset are sampled at a rate of 5 frames/s from videos, while others are captured from desired relative poses. Professionals manually annotate all images.

Det-Fly covers a variety of scenes, including different viewpoints, background settings, relative distances, and lighting conditions. Specifically, Det-Fly features four types of environmental backgrounds: sky, urban, field, and mountain. Each background type occupies roughly the same proportion of the dataset (approximately 20%–30%). Regarding relative viewpoints, Det-Fly can be categorized into three types: front view, top view, and bottom view. The data distribution for these three viewpoints is 36.4% (front view), 32.5% (top view), and 31.1% (bottom view).

4) DUT: The DUT Anti-UAV dataset is divided into training, testing, and validation sets. Specifically, the detection dataset contains a total of 10 000 images, with 5200 images in the training set, 2200 images in the testing set, and 2600 images in the validation set. Considering that each image may contain multiple objects, the total number of objects in the dataset is 10 109, with 5243 objects in the training set, 2245 in the testing set, and 2621 in the validation set. To enrich the diversity of objects and prevent model overfitting, the DUT dataset includes more than 35 types of MAV.

B. Implementation Details

1) Training Phase: We conduct the training of our DRNet model utilizing publicly datasets from NPS-Drone [3], TIBNet [26], Det-Fly [31], and DUT [32], respectively. The model is trained on a single GTX 3090 GPU employing the stochastic gradient descent (SGD) optimizer, and the training iterations are 200k. The initial learning rate is set at $\eta_{\text{initial}} = 1.3 \times 10^{-3}$. The learning rate schedule includes reductions by a factor of ten at both 80% and 90% of the planned iterations. We set the *weight decay* = 0.0005 to prevent overfitting and used a *momentum* = 0.949 to accelerate the optimization process. Regarding data augmentation, we do not apply rotation (angle = 0), but set the saturation = 1.5, exposure = 1.5, and hue = 0.1.

2) Test Phase: The method trained on TIBNet employs the TITAN XP to measure latency. Due to this device being somewhat outdated and difficult to procure, based on DTD-YOLOv4-Tiny [25] comparisons, the performance of the TITAN XP is equivalent to that of the RTX 2080Ti. Consequently, we also opt to measure latency on the RTX 2080Ti. Similarly, the latency tests for DUT are conducted on the RTX 3060, which offers performance that intersects that of the RTX 2080 Super. Each billion floating point operations (BFLOPs) and latency test is conducted using the same input resolution.

TABLE II
BENCHMARKING RESULTS ON NPS-DRONES

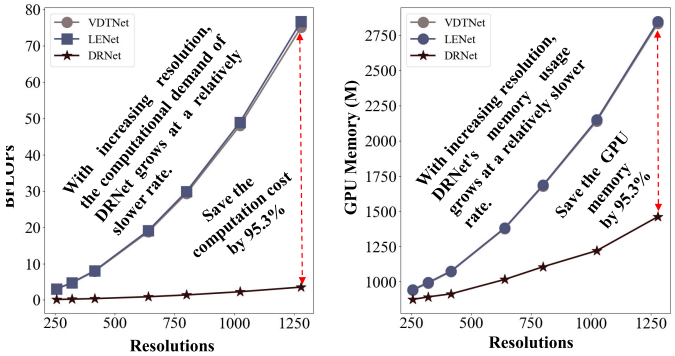
Model	Pre. (%)↑	Rec. (%)↑	F1 (%)↑	$mAP_{0.5}$ (%)↑	FPS	Model size↓
SCRDet-H [33]	81	74	77	65	-	400M
SCRDet-R [33]	79	71	75	61	-	400M
FCOS [34]	88	84	86	83	-	-
Mask-RCNN [35]	66	91	76	89	17.6	-
LENet [5]	88	91	90	91	-	4.5M
MEGA [36]	88	82	85	83	-	-
SLSA [37]	47	67	55	46	-	-
De-DETR [38]	-	-	-	76	10.7	-
VisTR [39]	-	-	-	66	1.6	218M
Dogfight [22]	92	91	92	89	1.0	-
YOLOv5-tph [40]	-	-	-	92	25.0	-
VDTNet [4]	91	92	92	92	-	3.9M
TransVisDrone [13]	92	91	92	95	24.6	939M
DRNet	**82**	**94**	**88**	**95**	**24.8**	**309K**

C. Comparison With Prior Works

1) NPS-Drone: In the experiments, the training process took approximately ten days. DRNet exhibits significant advantages across various performance metrics, as outlined in Table II, particularly excelling in model size, inference speed, and accuracy.

First, DRNet achieves excellent results in recall, $F1$-score, and mean average precision (mAP), attaining values of 94%, 88%, and 95%, respectively, positioning it among the top-performing models. In comparison, newer models such as LENet and VDTNet achieve comparable results to DRNet in precision and recall but fall slightly short in mAP, reaching only 91% and 92%, respectively, against DRNet's 95%. Although Dogfight [22] holds a slight edge in precision and $F1$-score, its mAP is six points lower than DRNet's, and it operates at an inference speed of just 1 frames/s on an RTX A6000, making it impractical for deployment on low-end edge devices.

A key advantage of DRNet is its lightweight model structure. The DRNet model occupies only 309 kB, while other high-performance models, such as TransVisDrone and VisTR,



(a) Computational Complexity Comparison (b) GPU Memory Comparison

Fig. 5. Comparison of the computational complexity and memory usage between DRNet, LENet, and VDTNet. (a) Comparison of computational complexity. (b) Comparison of memory usage. The horizontal axis represents the input resolution, while the vertical axis indicates BFLOPs and GPU memory usage, respectively.

have model sizes of 939 and 218 MB, respectively. Although the recent model VDTNet is relatively compact at 3.9 MB, DRNet remains one-tenth of its size while surpassing it in accuracy by 3% (95% versus 92%). This demonstrates that DRNet delivers exceptional performance while retaining a substantial efficiency advantage, rendering it particularly well-suited for resource-constrained application scenarios.

Furthermore, DRNet's inference speed of 24.8 frames/s is outstanding compared to other models, nearly matching YOLOv5-tph's 25.0 frames/s, thus meeting the requirements for real-time detection. YOLOv5-tph [40] and De-DETR [38] have model sizes of 119 and 389 MB, respectively, both much larger than DRNet's 309 kB. To the authors' knowledge, YOLOv5-tph exhibits a substantial memory requirement of up to 4.7 GB and a computational complexity reaching 557 BFLOPs, which presents significant challenges for deployment on resource-limited devices. Consequently, DRNet achieves superior performance and exhibits remarkable efficiency, positioning it as a highly competitive model for NPS-Drones scenarios.

Compared to the lightweight models VDTNet and LENet, the proposed DRNet demonstrates substantial advantages in terms of computational complexity and memory usage. As illustrated in Fig. 5(a), DRNet exhibits significantly lower computational complexity than the two methods, with this difference becoming more pronounced when processing high-resolution images. Fig. 5(b) further indicates that DRNet requires considerably less GPU memory for large-sized images. Notably, DRNet not only achieves higher detection accuracy, but also demonstrates memory efficiency and reduced complexity when handling high-resolution images. For instance, with images at a resolution of 1280×1280, DRNet reduces computational complexity by 95.3% and memory usage by 50% compared to VDTNet and LENet. This substantial performance improvement enables DRNet to process large images efficiently while maintaining low computational and memory requirements, making it a practical and cost-effective solution for real-world applications. Furthermore, as depicted in the figures, LENet and VDTNet show marked increases in memory usage and computational complexity, underscoring DRNet's ability to achieve accurate MAV detection with lower computational and memory demands.

TABLE III
BENCHMARKING RESULTS ON TIBNET

Method	Backbone	$mAP_{0.5}$ (%)↑	Latency (ms)↓	Model size↓
Faster RCNN [9]	ResNet50	87.2	217	333.5M
Faster RCNN [9]	MobileNet	67.5	125	162.5M
Cascade R-CNN [17]	MobileNet	78.0	164	384.9M
YOLOv3 [41]	DarkNet53	84.9	66	234.1M
YOLOv4 [30]	CSPDarkNet53	86.0	19	256.0M
EXTD [27]	MobileFaceNet	85.1	274	696.9KB
TIBNet [26]	TIB-Net	89.2	290	697.0KB
YOLOv4-Tiny [30]	-	78.5	-	23.0M
DTD-V4-Tiny [25]	-	85.1	-	1.4M
DRNet (ours)	**-**	**89.5**	**93**	**309KB**

2) TIBNet: In the original study [26], the number of training iterations on this dataset is 300k. However, we only train for 200k iterations. The training process takes approximately ten days for our network. On the TIBNet dataset [26], which is tailored for small object detection, the proposed DRNet is benchmarked against other state-of-the-art models in terms of accuracy (mAP), latency, and model size. DRNet consistently achieves the smallest model size while maintaining superior accuracy in Table III.

In detail, compared to the smallest model, EXTD, DRNet is less than half the size (309 kB versus 699 kB), surpasses it by 4% in accuracy (89.5% versus 85.1%), and achieves nearly three times faster inference speed (93 ms versus 274 ms). Additionally, when measured against the most accurate small model, TIBNet [26], DRNet demonstrates superior performance across all three evaluation metrics (89.5% versus 89.2%, 93 ms versus 290 ms, and 309 kB versus 697 kB).

Compared to traditional two-stage models, DRNet exhibits outstanding performance across various evaluation metrics. For example, when compared with Faster R-CNN using ResNet50 as the backbone, DRNet reduces the number of parameters by 99.9%, achieves double the speed, and enhances accuracy by 2% (89.5% versus 87.2%). Even in comparison with Faster R-CNN employing the lightweight MobileNet backbone, DRNet still conserves parameters by 99.8% (309 kB versus 162.5 MB) and remarkably boosts accuracy by 32.0% (89.5% versus 67.5%). Moreover, the largest two-stage model, Cascade R-CNN, with a model size of 384.9 MB, falls significantly short in terms of accuracy (89.5% versus 78.0%).

Compared to YOLOv4 and YOLOv3, although our DRNet shows a slight disadvantage in speed, DRNet continues to excel. DRNet improves mAP by 3.5% (89.5% versus 86%), while its model size is only one-thousandth that of YOLOv4 (309 kB versus 256 MB). Similarly, DRNet maintains a considerable advantage over the single-stage YOLOv3 model in both accuracy and model size.

As illustrated in Fig. 6, DRNet achieves an impressive balance between accuracy and speed, indicating that it not only detects small aerial vehicles with greater speed and accuracy, but also minimizes memory usage. DRNet surpasses existing state-of-the-art models across multiple metrics, demonstrating exceptional performance and efficient detection capabilities, particularly in resource-constrained environments.

3) Det-Fly: The DRNet method is comprehensively compared to other state-of-the-art models on the Det-Fly dataset,

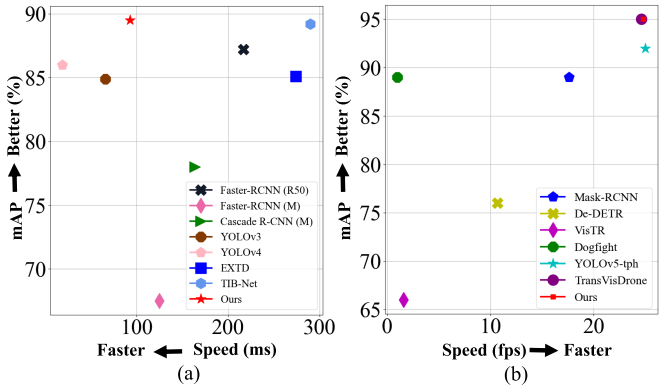


Fig. 6. Tradeoff performance for accuracy (mAP) versus speed. (a) NPS–Drone. (b) TIBNet. Our method DRNet superiors other methods in tradeoff performance. Please zoom in for the best view.

TABLE IV
BENCHMARKING RESULTS ON DET-FLY

Method	Image size	$mAP_{0.5}$ (%) ↑	Model size ↓	FLOPs (B)↓
Cascade R-CNN [17]	[640,640]	79.4	552.6 M	-
RetinaNet [42]	[600,600]	77.9	80.0 M	-
RefineDet [43]	[320,320]	69.5	78.1 M	-
FPN [44]	[600,600]	78.7	97.7 M	-
Faster R-CNN [9]	[1000,600]	70.5	333.5 M	-
Grid R-CNN [45]	[600,600]	82.4	493.0 M	-
SSD512 [46]	[416,416]	78.7	96.3 M	-
DRNet (ours)	[640,640]	86.6	309KB	0.9

TABLE V
BENCHMARKING RESULTS ON DUT

Model	Backbone	$mAP_{0.5:0.95}$ (%)↑	Latency (ms)↓
YOLOX [47]	ResNet50	42.7	46.1
	ResNet18	40.0	18.6
DRNet (Ours)	-	**44.3**	**10.5**

evaluating accuracy (mAP), latency, model size, and FLOPs. The Det-Fly dataset, designed for small object detection, serves as a benchmark for assessing model performance in practical applications. DRNet demonstrates significant advantages across multiple dimensions.

Table IV presents the benchmarking results for various models on the Det-Fly dataset. DRNet showcases high accuracy, extremely low latency, model size, and computational complexity substantially smaller than those of competing models. Specifically, DRNet achieves an mAP of 86.6%, a latency of 53 ms, a model size of only 309 kB, and FLOPs of just 0.9 B. These results surpass several advanced models, including Cascade R-CNN, RetinaNet, and Faster R-CNN. Notably, Grid R-CNN, with a model size of 493.0 MB, attains an accuracy of 82.7%, which is lower than DRNet's 86.6%, underscoring DRNet's ability to significantly reduce model size and computational complexity while maintaining high accuracy. Furthermore, compared to the single-stage model SSD, DRNet reduces parameters by 99.7% and achieves nearly 20% higher accuracy.

4) DUT: YOLOX is a robust object detection method, known for its strong generalization capabilities and rapid processing on the DUT dataset. Table V presents a comparison between DRNet and YOLOX with various backbone networks. In the original DUT dataset article [32], latency is measured on an RTX 2080 Super GPU, whereas DRNet's latency is recorded on an RTX 3060 GPU, which has a lower performance profile than the RTX 2080 Super. Notably, the model size of ResNet18 is 21.4 MB, indicating that the YOLOX (ResNet18) model size exceeds 21.4 MB, and similarly, the YOLOX (ResNet50) model size exceeds 46.8 MB.

Compared to YOLOX (ResNet18), DRNet achieves a four-point accuracy improvement (44.3% versus 40.0%), while conserving at least 98.6% of the parameters and offering faster inference (10.5 ms versus 18.6 ms). In comparison to YOLOX (ResNet50), DRNet is four times faster in inference speed (10.5 ms versus 46.1 ms) and achieves nearly a two-point accuracy improvement (44.3% versus 42.7%) while reducing parameters by 99.4%. These results underscore DRNet's capability to provide high efficiency and performance, particularly in resource-constrained settings.

TABLE VI
ABLATION STUDY 1

Experiments	mAP	Model size	FLOPs
a1	85.2%	296KB	0.962B
a2	85.7%	292KB	0.872B
a3	84.8%	249KB	2.37B
a4	81.4%	2.7M	2.61B
Baseline	**86.4%**	**296KB**	**1.0B**

D. Ablation Study

1) Part One of the Ablation Study: To demonstrate the rationale and superiority of the methods designed in this study, an ablation study is conducted using a baseline network structure that excludes channel attention calculations but includes residual connections between SER modules. The ablation study consists of two parts: the first part focuses on the Det-Fly dataset, while the second part evaluates performance across four datasets. The first part of the ablation study comprises the following experiments:

1) replacing concatenation in the DSC module with an addition operation;
2) removing the residual connections between SER modules (not shown here as residual connections are ultimately removed);
3) removing both the residual connections between SER modules and one SER module lacking integrated channel attention calculation;
4) adding DSC after the existing DSC in the network.

The results from the first phase of the ablation study are presented in Table VI. When concatenation in the DSC module is replaced with an addition operation, the computational cost (BFLOPs) slightly decreases without affecting model size, but accuracy drops by 1.2%. This drop is due to the reduction in the number of channels from the addition operation, which lowers the parameter count. However, since the baseline network has relatively few channels, both concatenation and addition have minimal impact on model size. Nevertheless, concatenation better preserves features, yielding a more significant accuracy benefit.

Eliminating the residual connections between the SER modules decreases the number of network layers, which, in turn,

TABLE VII
ABLATION STUDY 2

Experiments	Model size	mAP1	mAP2	mAP3	mAP4
Baseline	296KB	86.4%	93%	88.3%	42.6%
+Channel attention computation	294KB	84.5%	94%	88.9%	40.7%
+Spatial attention computation	309KB	**86.6%**	**95%**	**89.5%**	**44.3%**
+Reorg computation	246KB	85.4%	94%	89.3%	42.4%

reduces the parameter count. Consequently, the model size shrinks by 4 kB, and the computational requirements fall to 0.872 BFLOPs. However, this reduction also diminishes the model's representation capacity, resulting in a 0.7% drop in accuracy.

When an SER module without channel attention calculation and its residual connection are removed, the parameter count and network structure are notably simplified. This simplification severely impacts the model's representation power, leading to substantial reductions in both model size and accuracy. Additionally, the removal of these layers causes an increase in the feature map size output by the model, which raises the computational burden required to process these larger feature maps, resulting in BFLOPs rising to 2.37.

As shown in the last row of Table VI, since the number of channels in the DSC module can reach up to 256, adding an additional DSC layer significantly increases the model size without proportionately enhancing its representational capacity, leading to a noticeable decline in accuracy. This finding suggests that reintroducing another DSC module can result in over-representation. While adding an extra DSC has a limited impact on the output feature map, the computational cost remains high at 2.61 BFLOPs. Therefore, this work utilizes only a single DSC layer in DRNet.

Although the baseline model performs well on the Det-Fly dataset, its performance is suboptimal on other datasets, such as NPS-Drone, TIBNet, and DUT. Consequently, a combined ablation study on these four datasets is conducted to determine the model with the best overall performance, which is then presented as the final detection method, referred to as DRNet in this work.

2) Part Two of the Ablation Study: The results from the second phase of the ablation study are presented in Table VII, where mAP1, mAP2, mAP3, and mAP4 denote the model's detection accuracy on the Det-Fly, NPS-Drone, TIBNet, and DUT datasets, respectively. In this experiment, channel attention calculation is integrated into the residual network to form the SER module. To reduce parameters, the residual connections between SER modules, which contain convolution and Dropout layers, are removed due to the additional convolution operations introduced by channel attention.

This ablation experiment improves detection accuracy on the NPS-Drone and TIBNet datasets though accuracy decreases on the Det-Fly and DUT datasets. According to the target size distribution analysis shown in Fig. 4, the TIBNet and NPS-Drone datasets predominantly consist of small targets, while the Det-Fly and DUT datasets contain relatively fewer small targets. Consequently, channel attention calculation proves more effective for enhancing the detection accuracy of small targets.

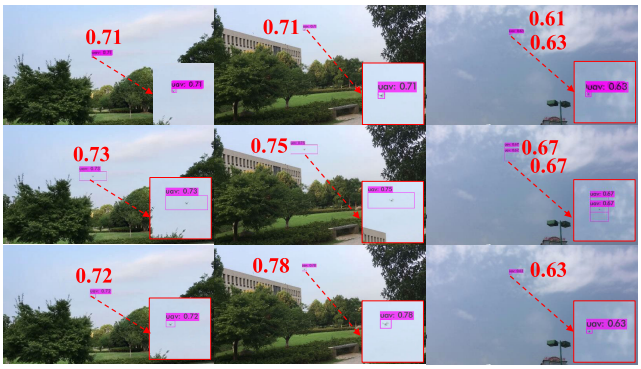


Fig. 7. Visualization of the ablation study models. Each column represents the same scene, and each row shows the detection results for a specific method. The red rectangles indicate magnified detection targets. The first and second rows of the third column mistakenly detect a single MAV as two separate objects.

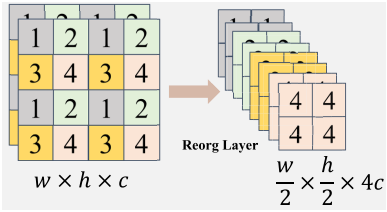


Fig. 8. Reorg layer information processing principle.

To improve detection capabilities on the TIBNet and NPS-Drone datasets and enhance performance on the Det-Fly and DUT datasets, further experiments are conducted. Since channel attention strengthens the model's representation capability but is less effective for target localization, a spatial attention calculation is introduced in the feature fusion section to suppress redundant information and improve localization accuracy. As indicated by the results in Table VII, the introduction of spatial attention calculation enhances detection accuracy and localization quality across all datasets.

Fig. 7 presents the detection visualization results for the baseline model, the model with integrated channel attention calculation, and the model incorporating the spatial attention calculation within identical scenes. Each column represents the same scene, while each row displays the detection visualization results corresponding to each model.

The baseline model exhibits low detection confidence and limited modeling capability. While channel attention calculation enhances the model's representational capacity, its localization performance remains inadequate, resulting in considerable discrepancies between predicted bounding boxes and actual target boundaries. By introducing spatial attention calculation, the model further improves its representational capacity and enhances localization accuracy, resulting in bounding boxes that are better aligned with target boundaries and a substantial increase in detection confidence. These visualizations further validate the effectiveness of the strategies employed in this work.

Additionally, this experiment explores the use of the Reorg layer [48]. As shown in Fig. 8, the Reorg operation supports downsampling while preserving complete information transfer. Specifically, the Reorg layer extracts alternate features from the input tensor and concatenates them along the channel

(a) (b) (c)

Fig. 9. Detection results under various challenging conditions. The cropped images from the top right of the second and third columns are for the best views of the targets, whereas the cropped images in the bottom right corner depict the target and its background. Please zoom in for the best view. (a) Small MAVs. (b) Low-light scenes. (c) Camouflage scenes.

dimension, thereby enhancing the depth of the feature representation. Following the Reorg layer's transformation, the spatial resolution of the feature map is reduced by half, while the number of channels increases to four times the initial count. To reduce the model size, the original 256-channel convolution layer is decreased to eight channels. The fourth row of Table VII presents the results of the Reorg operation, demonstrating a significant reduction in model size with only a minor decrease in detection accuracy.

In conclusion, the comprehensive approach outlined in the third row of Table VII is selected as the final detection method, namely, DRNet.

E. MAV Detection Evaluation Under Various Real-World Conditions

As illustrated in Fig. 9, we evaluate DRNet across various scenarios, with each column representing the same scenario. Specifically, the first column depicts the small MAV scenario, the second column features small MAVs under low-light conditions, and the third column represents camouflage scenes. Enlargements of the objects are shown in the red boxes within the first column. In the second and third columns, the top right corner contains a cropped image of the detected target, while the bottom right corner shows a screenshot of the target within its environment. Camouflage, a common survival and predatory tactic in the biological world, aims to blend an organism's body color with its surroundings to achieve invisibility, making it difficult for predators or prey to detect. Similarly, in MAV detection, camouflage scenarios arise where the MAV blends seamlessly with its environment due to its color or external factors, such as lighting or weather complicating its detection. *To the authors' knowledge, this is the first that MAV detection in camouflage scenarios has been considered.*

In small MAV scenarios, the detection system is often positioned considerably from the MAV, resulting in extremely small pixel representations of the detected targets. Such conditions pose significant challenges for feature extraction. However, our method can detect MAVs, thanks to incorporating channel attention within our SER architecture, which focuses on the detected targets. The extracted features are fused with convolutional features. Furthermore, our use of the spatial attention calculation helps the network pay attention to the positional information of the MAV, which proved to be an effective strategy for detecting small MAVs.

In low-light and camouflage scenarios, DRNet continues to detect MAVs, confirming the feasibility and effectiveness of our approach. Especially in camouflage situations, such as in

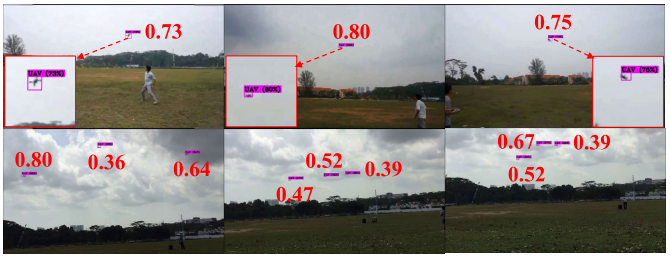


Fig. 10. Detection results on edge-computing device. The red rectangles indicate magnified detection targets.

the first and fourth rows, where the MAV blends extremely well with its surroundings, it becomes difficult to discern the target even from the bottom-right corner screenshots with the naked eye. However, our method still detects the MAV without false positives.

F. Actual Experiments on Edge-Computing Device

In this section, DRNet is deployed on an NVIDIA Orin NX device equipped with a 16-GB GPU and eight CPUs for real-world MAV detection scenarios. To comprehensively assess system performance, metrics such as detection speed, the operating temperature of the Orin NX, and power consumption were monitored throughout the experiment, with detection results displayed on a computer terminal.

As depicted in Fig. 10, the video input resolution is set at 1920 × 1080. During real-world MAV detection testing, DRNet achieves a throughput of 14.4 frames/s. The sensor temperature generally remains below 49.5 °C (acceptable operating range: −25 °C to 105 °C), with a power consumption of 2.6 W. Fig. 10 illustrates detection visualizations from two video sequences, where each row corresponds to the detection visualizations for the same scene. The scene in the first row is overcast, and the detected MAV is a handmade model rather than a commercial MAV, which results in morphological differences compared to commercial MAVs. During detection, the MAV is located farther from the detection system, leading to fewer pixels and a blurred appearance. The scene in the second row shows three MAVs flying laterally in a horizontal formation. Compared to the first row, this scene involves a greater distance and more detection targets. These detection results demonstrate that, even with high-resolution video input, the proposed method performs fast and accurate MAV detection effectively, affirming the portability and reliability of DRNet.

Further, DRNet is compared with new methods, VDTNet and LENet, with all models configured to an input size of 640 × 640. The evaluation metrics include power consumption, inference speed, and temperature. As shown in Table VIII, it is observed that the proposed method, DRNet, exhibits lower power consumption, achieves inference speeds twice as fast as the other two methods (14.4 frames/s versus 7.2 frames/s), and operates at a lower temperature.

Additionally, to further test model performance, DRNet is deployed on a CPU for real-world MAV detection. Results indicate a throughput of 2.7 frames/s for DRNet on the CPU, which is nearly ten times faster than the other two methods (2.7 frames/s versus 0.2 frames/s). Interestingly, DRNet exhibited higher CPU throughput than a model without integrated attention calculation. This preliminary experiment suggests that although attention calculation may typically increase the computational burden on GPUs, they do not negatively impact processing speed on the CPU; rather, they facilitate improved detection efficiency.

TABLE VIII
EDGE DEVICE EXPERIMENT

Methods	Power (W)	Speed$_{gpu}$ (FPS)	Speed$_{cpu}$ (FPS)	Temperature (°C)
VDTNet	3.1	7.4	0.3	50.9
LENet	3.2	7.2	0.2	51.0
DRNet (Ours)	**2.6**	**14.4**	**2.7**	**49.5**

Overall, DRNet's edge deployment on the NVIDIA Orin NX device and the testing results on the CPU demonstrate the method's efficiency and practicality in real-world applications. DRNet can reliably and rapidly execute MAV detection tasks on GPU or CPU platforms, verifying its broad applicability and superior performance across different computational environments.

V. CONCLUSION

In this study, we have proposed the miniature MAV detection network (DRNet) to accurately detect MAVs with limited computational costs and memory resources. The detection performance of different methods for MAV detection was compared. Subsequently, multiple challenging scenarios testing and actual edge-computing device experiments validate the efficacy and efficiency of our method, respectively, and the following conclusions were drawn.

1) DRNet demonstrates efficacy and efficiency in MAV detection. DRNet achieves accuracy comparable to the transformer-based TransVisDrone [13], yet with a faster inference speed. Notably, the parameters are reduced by 99.97% compared to TransVisDrone. DRNet also demonstrates a better balance between accuracy and latency. This improvement in parameters coupled with high accuracy highlights the superiority of DRNet for MAV detection deployment on memory-constrained devices.
2) DRNet demonstrates low complexity and low memory usage when processing high-resolution images. Specifically, when the input resolution increases from 256 × 256 to 1280 × 1280, the computational cost (FLOPs) increases by no more than 4 billion, while the GPU memory usage increases by no more than 822 MB. In contrast, VDTNet and LENet [4], [5], with a model size of only 3.9 and 4.5 MB, experience increases of 72 billion in FLOPs and 1894M in GPU memory usage. At a resolution of 1280 × 1280, DRNet reduces computational complexity by 95.3% and GPU memory usage by 50% compared to VDTNet and LENet.
3) The feasibility and portability of DRNet have been demonstrated through MAV testing in scenarios involving small targets, low light, and camouflage. DRNet is capable of detecting MAVs in these three scenarios with

high confidence. Deploying DRNet on the Orin NX GPU for MAV detection experiments achieved a detection speed of 14.4 frames/s, demonstrating its capability for fast detection. Additionally, in deployment experiments utilizing only the CPU, DRNet achieved a detection speed of 2.7 frames/s. Although lower, this still indicates that DRNet can provide a reasonable level of detection efficiency without GPU support. These results highlight DRNet's potential for real-time MAV detection on edge-computing devices, particularly under constrained computational resources.

In summary, DRNet introduced in this study exhibits notable advantages in MAV detection under memory-constrained conditions. Given the relatively small size of the DRNet model, there is still room for improvement in inference speed. The proposed method demonstrates the feasibility of detecting objects in low-light, small-object, and camouflage scenarios. However, its performance tends to degrade under adverse weather conditions, such as rain, snow, or fog. Future research endeavors could explore network architectures more suitable for deployment on edge devices to further enhance detection efficiency and expand its deployment scope. Further exploration of image enhancement techniques to support object detection methods would also contribute to achieving all-weather MAV detection.

REFERENCES

[1] H. C. Kumawat, M. Chakraborty, and A. A. B. Raj, "DIAT-RadSATNet—A novel lightweight DCNN architecture for micro-Doppler-based small unmanned aerial vehicle (SUAV) targets' detection and classification," *IEEE Trans. Instrum. Meas.*, vol. 71, pp. 1–11, 2022.

[2] X. Zhou et al., "ADMNet: Anti-drone real-time detection and monitoring," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Oct. 2023, pp. 3009–3016.

[3] J. Li, D. H. Ye, T. Chung, M. Kolsch, J. Wachs, and C. Bouman, "Multi-target detection and tracking from a single camera in unmanned aerial vehicles (UAVs)," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Aug. 2016, pp. 4992–4997.

[4] X. Zhou, G. Yang, Y. Chen, L. Li, and B. M. Chen, "VDTNet: A high-performance visual network for detecting and tracking of intruding drones," *IEEE Trans. Intell. Transp. Syst.*, vol. 25, no. 8, pp. 9828–9839, Aug. 2024.

[5] X. Zhou, L. Li, and B. M. Chen, "LENet: Lightweight and effective detector for aerial object," *Unmanned Syst.*, vol. 12, no. 6, pp. 1105–1121, Nov. 2024.

[6] S. Chen et al., "TinyDet: Accurately detecting small objects within 1 GFLOPs," *Sci. China Inf. Sci.*, vol. 66, no. 1, pp. 1–2, Jan. 2023.

[7] Z. Qin et al., "ThunderNet: Towards real-time generic object detection on mobile devices," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2019, pp. 6718–6727.

[8] R. J. Wang, X. Li, and C. X. Ling, "Pelee: A real-time object detection system on mobile devices," in *Proc. Adv. Neural Inf. Process. Syst. (NIPS)*, vol. 31, Jan. 2018, pp. 1–17.

[9] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks," 2015, *arXiv:1506.01497*.

[10] T. Ye, W. Qin, Z. Zhao, X. Gao, X. Deng, and Y. Ouyang, "Real-time object detection network in UAV-vision based on CNN and transformer," *IEEE Trans. Instrum. Meas.*, vol. 72, pp. 1–13, 2023.

[11] J. Hu, L. Shen, and G. Sun, "Squeeze-and-excitation networks," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 7132–7141.

[12] X. Zhu, D. Cheng, Z. Zhang, S. Lin, and J. Dai, "An empirical study of spatial attention mechanisms in deep networks," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, Oct. 2019, pp. 6688–6697.

[13] T. Sangam, I. R. Dave, W. Sultani, and M. Shah, "TransVisDrone: Spatio-temporal transformer for vision-based drone-to-drone detection in aerial videos," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2023, pp. 6006–6013.

[14] X. Zhou, B. Zhao, G. Yang, J. Zhang, L. Li, and B. M. Chen, "SANet: Small but accurate detector for aerial flying object," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2024, pp. 17882–17888.

[15] A. G. Howard et al., "MobileNets: Efficient convolutional neural networks for mobile vision applications," 2017, *arXiv:1704.04861*.

[16] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 770–778.

[17] 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.

[18] M. Tan, R. Pang, and Q. V. Le, "EfficientDet: Scalable and efficient object detection," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 10781–10790.

[19] Y. Cai et al., "YOLObile: Real-time object detection on mobile devices via compression-compilation co-design," in *Proc. AAAI Conf. Artif. Intell.*, 2021, vol. 35, no. 2, pp. 955–963.

[20] Z. Li, C. Peng, G. Yu, X. Zhang, Y. Deng, and J. Sun, "Light-head R-CNN: In defense of two-stage object detector," 2017, *arXiv:1711.07264*.

[21] H. Dong, J. Liu, C. Wang, H. Cao, C. Shen, and J. Tang, "Drone detection method based on the time-frequency complementary enhancement model," *IEEE Trans. Instrum. Meas.*, vol. 72, pp. 1–12, 2023.

[22] M. W. Ashraf, W. Sultani, and M. Shah, "Dogfight: Detecting drones from drones videos," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2021, pp. 7063–7072.

[23] Q. Cheng, X. Li, B. Zhu, Y. Shi, and B. Xie, "Drone detection method based on MobileViT and CA-PANet," *Electronics*, vol. 12, no. 1, p. 223, Jan. 2023.

[24] Y. Lv, Z. Ai, M. Chen, X. Gong, Y. Wang, and Z. Lu, "High-resolution drone detection based on background difference and SAG-YOLOv5s," *Sensors*, vol. 22, no. 15, p. 5825, Aug. 2022.

[25] R. Jiang, Z. Ye, Y. Peng, G. Xie, and H. Du, "Lightweight target detection algorithm for small and weak drone targets," *Prog. Laser Optoelectron.*, vol. 59, no. 8, pp. 109–120, 2022.

[26] H. Sun, J. Yang, J. Shen, D. Liang, L. Ning-Zhong, and H. Zhou, "TIB-Net: Drone detection network with tiny iterative backbone," *IEEE Access*, vol. 8, pp. 130697–130707, 2020.

[27] Y. Yoo, D. Han, and S. Yun, "EXTD: Extremely tiny face detector via iterative filter reuse," 2019, *arXiv:1906.06579*.

[28] H. Fang, L. Ding, L. Wang, Y. Chang, L. Yan, and J. Han, "Infrared small UAV target detection based on depthwise separable residual dense network and multiscale feature fusion," *IEEE Trans. Instrum. Meas.*, vol. 71, pp. 1–20, 2022.

[29] D. Misra, "Mish: A self regularized non-monotonic activation function," 2019, *arXiv:1908.08681*.

[30] A. Bochkovskiy, C.-Y. Wang, and H.-Y. Mark Liao, "YOLOv4: Optimal speed and accuracy of object detection," 2020, *arXiv:2004.10934*.

[31] Y. Zheng, Z. Chen, D. Lv, Z. Li, Z. Lan, and S. Zhao, "Air-to-air visual detection of micro-UAVs: An experimental evaluation of deep learning," *IEEE Robot. Autom. Lett.*, vol. 6, no. 2, pp. 1020–1027, Apr. 2021.

[32] 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.

[33] X. Yang et al., "SCRDet: Towards more robust detection for small, cluttered and rotated objects," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, Oct. 2019, pp. 8232–8241.

[34] 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. 9627–9636.

[35] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask R-CNN," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Oct. 2017, pp. 2961–2969.

[36] Y. Chen, Y. Cao, H. Hu, and L. Wang, "Memory enhanced global-local aggregation for video object detection," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 10337–10346.

[37] G. Jocher. (May 2020). *YOLOv5 By Ultralytics*. [Online]. Available: https://github.com/ultralytics/yolov5

[38] X. Zhu, W. Su, L. Lu, B. Li, X. Wang, and J. Dai, "Deformable DETR: Deformable transformers for end-to-end object detection," 2020, *arXiv:2010.04159*.

[39] Y. Wang et al., "End-to-end video instance segmentation with transformers," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2021, pp. 8741–8750.
[40] X. Zhu, S. Lyu, X. Wang, and Q. Zhao, "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)*, Oct. 2021, pp. 2778–2788.
[41] J. Redmon and A. Farhadi, "YOLOv3: An incremental improvement," 2018, *arXiv:1804.02767*.
[42] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, "Focal loss for dense object detection," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Oct. 2017, pp. 2980–2988.
[43] S. Zhang, L. Wen, X. Bian, Z. Lei, and S. Z. Li, "Single-shot refinement neural network for object detection," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 4203–4212.
[44] T. Y. Lin, P. Dollàr, R. Girshick, K. He, B. Hariharan, and S. Belongie, "Feature pyramid networks for object detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jul. 2017, pp. 2117–2125.
[45] X. Lu, B. Li, Y. Yue, Q. Li, and J. Yan, "Grid R-CNN," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 7363–7372.
[46] W. Liu et al., "SSD: Single shot multibox detector," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, Oct. 2016, pp. 21–37.
[47] Z. Ge, S. Liu, F. Wang, Z. Li, and J. Sun, "YOLOX: Exceeding YOLO series in 2021," 2021, *arXiv:2107.08430*.
[48] J. Redmon and A. Farhadi, "YOLO9000: Better, faster, stronger," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 7263–7271.

Xunkuai Zhou (Member, IEEE) is currently pursuing the Ph.D. degree in control science and engineering with Tongji University, Shanghai, China.

He is also involved in research works as a Visiting Ph.D. Student with the Department of Mechanical and Automation, The Chinese University of Hong Kong, Hong Kong, China. His current research interests include model compression, object detection, object tracking, and the application of unmanned systems in smart architecture.

Bingxin Han is currently pursuing the Ph.D. degree with The Chinese University of Hong Kong, Hong Kong, China.

Her current research interests include parsing city-scale scenes into compact 3-D models and evaluating the condition of the building.

Li Li (Member, IEEE) received the Ph.D. degree from Shenyang Institute of Automation, Chinese Academy of Science, Shenyang, China, in 2003.

She joined Tongji University, Shanghai, China, in 2003, where she is currently a Professor with the Department of Control Science and Engineering. Her research interests include data-driven modeling and optimization, and computational intelligence.

Jie Chen (Fellow, IEEE) received the B.Sc., M.Sc., and Ph.D. degrees in control theory and control engineering from Beijing Institute of Technology, Beijing, China, in 1986, 1996, and 2001, respectively.

He was the President of Tongji University, Shanghai, China, from 2018 to 2023. He is currently a Professor with the Control Science and Engineering, Beijing Institute of Technology and Tongji University, where he serves as the Director of the National Key Laboratory of Autonomous Intelligent Unmanned Systems (KAIUS). His research interests include complex systems, multiagent systems, multiobjective optimization and decision, and constrained nonlinear control.

Ben M. Chen (Fellow, IEEE) was an Assistant Professor with the Department of Electrical Engineering, State University of New York at Stony Brook, Stony Brook, NY, USA, from 1992 to 1993. He was a Provost's Chair Professor with the Department of Electrical and Computer Engineering, National University of Singapore (NUS), Singapore. He joined The Chinese University of Hong Kong (CUHK), Hong Kong, in 2018, where he is currently a Professor of mechanical and automation engineering. He has authored or co-authored over 100 journal articles and conference papers, and a dozen research monographs in control theory and applications, unmanned systems, and financial market modeling. His current research interests include unmanned systems and their applications.

Dr. Chen is a fellow of the Academy of Engineering, Singapore. He had served on the editorial boards of a dozen international journals, including *Automatica* and IEEE TRANSACTIONS ON AUTOMATIC CONTROL. He is currently serving as the Editor-in-Chief for *Unmanned Systems*.