Unmanned Systems, Vol. 12, No. 6 (2024) 1105–1121

DOI: 10.1142/S2301385025500384

UNMANNED SYSTEMS
https://www.worldscientific.com/worldscinet/us

LENet: Lightweight and Effective Detector for Aerial Object

Xunkuai Zhou *,†,‡, Li Li †,§, Ben M. Chen †,¶

*School of Electronics and Information Engineering,
Tongji University, Shanghai, P. R. China

†Department of Mechanical and Automation Engineering,
the Chinese University of Hong Kong,
Hong Kong, P. R. China

Aerial object detection is crucial in various computer vision tasks, including video monitoring, early warning systems, and visual tracking. While current methods can accurately detect normal-sized objects, they face challenges distinguishing small objects from cluttered backgrounds. Developing methods that can be deployed on edge devices to achieve fast, accurate, and energy-efficient performance is also an urgent challenge. This paper proposes a network for aerial object detection by incorporating an attention mechanism to enhance feature extraction and elevate the accuracy of aerial moving object detection. Additionally, we optimize the channel dimensions of the feature extraction framework, resulting in a reduction in model parameters, acceleration of inference speed, and alleviation of the computational burden. Ulteriorly, we optimize the Spatial Pyramid Pooling (SPP) module to enhance detection accuracy and processing speed. Inspired by the ResNet and RepVGG structures, we design a feature fusion module to combine early-extracted features, improving speed and accuracy. Based on the design mentioned above principles, we develop a neural network method with an impressively small model size of only 4.5 M. The proposed approach achieves state-of-the-art performance on five benchmark datasets. Besides its superior performance, our method demonstrates excellent throughput on edge computing devices. Experimental results show that even when running on low-performance computing devices, the CPU and GPU temperatures remain below 50°C and achieve a detection speed of 14.8 frames per second (fps) and power consumption of only 2.9 W. These findings suggest that a high-accuracy, low-power, low-latency, and low-memory footprint aerial object detection solution is achievable.

Keywords: Aerial object detection; energy-efficient; edge computing; deep learning.

US

1. Introduction

Object detection is one of the crucial research areas on computer vision and image processing, which guides the behavior of observers, including animals and robots. In complex dynamic environments, autonomous robot systems must detect object motion to understand movement intention, predict future paths, and react appropriately [1, 2]. For example, detecting potentially dangerous objects early in and far away would provide enough time for autonomous systems to respond timely, thereby maintaining or reinforcing their autonomous intelligence interaction and competition [3]. Existing methods perform well in detecting objects of regular sizes but exhibit subpar performance in detecting small objects in cluttered environments. This is primarily because small objects appear as faint patches on the frames, as observed from an observer or ontological standpoint, making it arduous to determine most of its visual features. As shown in Fig. 1, these aerial objects with a few pixels are almost invisible to the human eye. In addition, striking a balance between accuracy, computational consumption, and inference speed poses challenges in object detection algorithms. The practical implementation of such methods is hindered by their high-power consumption, demanding computational requirements, and substantial memory usage. For instance, when

Received 22 December 2023; Revised 27 March 2024; Accepted 27 March 2024; Published 8 May 2024. This paper was recommended for publication in its revised form by editorial board member, Jinqiang Cui.
Email Addresses: ‡*2010474@tongji.edu.cn*, §*lili@tongji.edu.cn*, ¶*xunkuaizhou@cuhk.edu.hk; bmchen@cuhk.edu.hk*
‡Corresponding author.

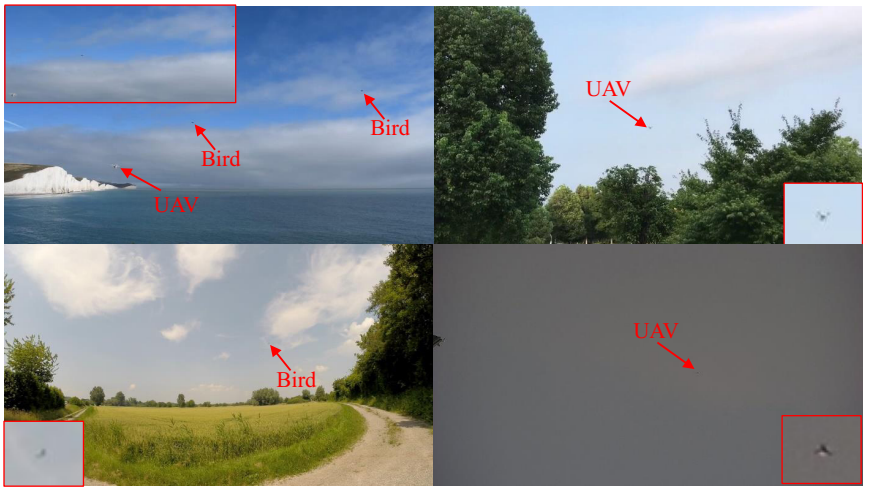


Fig. 1. Example of small aerial objects [4, 5]. In each subplot, enlargements of the objects are shown in the red boxes. The UAV and bird appear as dim speckles, only a few pixels in size, and most of their visual features are difficult to discern. In particular, they all show extremely low contrast against the cluttered backgrounds with illumination variations.

deploying object detection algorithms on resource-constrained devices like Unmanned Aerial Vehicles (UAVs) to monitor intruding drones captured by the camera, the slow processing of video frames becomes a significant limitation. Consequently, this delay may allow the intruding drone to escape detection before appropriate action can be taken. Moreover, the elevated power consumption of these algorithms results in increased device heat in a short period, leading to parameter drift and ultimately impacting the longevity of the video monitoring system.

Detecting airborne objects is challenging due to the noise generated during detection. The irregular shapes and sizes of aerial objects generate noise around them by the video codec standard, leading to misclassification or undetected targets. For example, surveillance videos may have inferior image quality due to limitations in sensor technology and sampling rates. The resultant blur silhouettes of airborne objects and noise increase false-positive rates. With the rapid development of deep learning from 2014 [6], based on the scheme of feature extraction, feature fusion, target classification, and localization, numerous deep learning-based object detection methods are utilized to reduce the influence caused by noise and improve the detection accuracy, such as Cascade RCNN [7], YOLO [8–10] and RetinaNet [11], etc., have achieved great progress in general object detection. However, it still struggles to detect small objects in cluttered environments because of the hard representation of learning from their tiny appearances. To address this issue, TPH-YOLOv5 [12] is proposed for drone-captured small object detection by adding the Transformer Prediction Heads (TPH). To further reduce the complexity and improve the detection speed of TPH-YOLOv5, the prediction heads of TPH-YOLOv5 are replaced with the cross-layer asymmetric transformer to propose TPH-YOLOv5++ [13] for further improving the detection accuracy. To adaptively detect both small objects and big objects, Jiang *et al.* propose GiraffeDet [14], and they argue that the main factor affecting detection performance is feature fusion rather than feature extraction. As such, they incorporate many convolutional layers in the feature fusion stage to improve detection accuracy. However, these methods need high computational resources. For example, the GPU memory footprint of TPH-YOLOv5++ [13] reaches 4.7 G, and the computational cost is up to 207.0 GFLOPs. It is worth noting that TPH-YOLOv5++ [13] achieves a relatively low frame processing speed of 11.9 frames per second (fps) on an NVIDIA RTX3090ti GPU. Additionally, even the smallest version of GiraffeDet, GiraffeDet-D7, still requires a computational power of 187 GFLOPs. The drawback of slow frame processing speed renders these computationally demanding algorithms unsuitable for deployment on low-performance devices, significantly impeding their practical applicability.

As shown in Fig. 1, one particularly important but very difficult aspect is accurately detecting targets in the presence of many similar objects. For example, in UAV applications, distinguishing between birds and UAVs is extremely challenging due to their physical similarity and similar motion patterns, increasing the difficulty of correctly classifying UAVs and birds. This interference from similar objects may result in erroneous detection results, affecting overall application effectiveness. The color variation of aerial objects can be significantly influenced by dynamic backgrounds, such as moving clouds and illumination changes, causing methods based on color extraction to exhibit poor performance. In particular, aerial detection

encompasses not only bottom-up detection but also top-down object detection. In contrast to the former with a singular background, the latter faces more complex ground backgrounds and lighting variations, further complicating aerial object detection [15]. The scale of aerial objects can vary greatly across frames, which presents a significant challenge to accurately detecting objects. This is because moving objects have no physical boundaries that constrain their movement in space.

Extensive research has been conducted to address the impact of complex backgrounds and similar objects on detection accuracy. Self-adaptive SOM-CNN [16] is a robust object detection method and decreases the false-positive rate in Dynamic backgrounds. The cross-scene background subtraction (CABs) algorithm [17] learns more discriminative semantic information between the foreground and the background, suppressing the problems from the dynamic background. Zhang *et al.* [18] detect moving objects by adding an explicit dynamic clutter component in the decomposition framework with realistic dynamic background modeling. Chen *et al.* [19] propose a new weighted Kernel Density Estimation (KDE) to build the Long-Term Background (LTB) and Short-Term Foreground (STF) models, respectively, which flexibly represent the long-term state and the short-term changes in a scene and improve the robustness of moving object detection. The other is to apply feature-based mechanisms. Tu *et al.* [20] propose a moving object detection method via ResNet-18 with an encoder-decoder structure to segment moving objects from complex scenes. However, these methods primarily focus on bottom-up environmental changes and do not explore the effectiveness of top-down detection. A drone-vs-bird dataset [4] comprising the drone and bird is proposed in the detection challenge at IEEE AVSS2021, putting a foundation for aerial moving object detection with similar objects. MFNet [21] is also an efficient drone and bird detection method using multi-feature fusion. However, further improvement of MFNet [21] in the localization and detection accuracy of the method would serve as an additional enhancement.

For object detection in aerial scenarios, the simultaneous presence of *complex backgrounds, interference from similar objects*, *high energy consumption*, and *slow reasoning speed* pose significant challenges. These challenges introduce complexities in accurately detecting moving objects in such scenarios. This work introduces a Lightweight and Energy-efficient method Network (LENet) for accurate aerial object detection to address these challenges. The contributions of this work can be summarized as follows:

- A lightweight yet energy-efficient method is designed for accurate aerial moving object detection by introducing the spatial attention mechanism, increasing the ability to discriminate in the dynamic background. Therefore, small aerial object detection could be accurately detected. This method optimizes each convolutional operation by employing fewer convolutional kernels, reducing the model's parameter count, faster inference speed, and decreasing computational resource consumption.
- Inspired by ResNet and RepVGG, they are commonly used for feature extraction. A feature fusion module is designed to retain semantic information and transfer more information to the latter layer. Additionally, we utilize pixel reorganization instead of convolutional operations for downsampling, improving detection effectiveness and efficiency.
- We conducted systematic ablation studies to validate the effectiveness of our designed approach. We demonstrate that our method surpasses state-of-the-art approaches through extensive comparative experiments on five challenging public datasets. Furthermore, deploying our method on edge-computing devices showcased its portability and feasibility in real-world applications. These findings collectively support achieving a high-accuracy, low-power, low-latency, and low-memory footprint solution for aerial object detection in practical applications is attainable.

The remainder of this paper is organized as follows. Related works are discussed in Sec. 2. Section 3 explains the model's theoretical basis and establishment process. Section 4 portrays the experimental results and performance evaluation. Section 5 draws the conclusions and perspective work.

2. Related Works

2.1. *Aerial object detection*

Detecting small targets in complex natural environments poses a significant challenge for autonomous robots. Much research has been done to solve this issue by introducing block segmentation for characterizing blocks into feature vectors and then estimating the moving direction based on feature vectors to measure the motions of blocks and filter out the major direction. Wang *et al.* [22] propose a novel method for accurate small aerial object detection in dynamic backgrounds. To enhance the capability to extract effective features of the aerial object with a few parameters. TIBNet [5] is proposed by introducing a spatial attention module circularly. Similarly, DTD-YOLOv4-Tiny [23] is proposed to efficiently detect aerial objects by recasting the features extraction network of YOLOv4-Tiny and optimizing the prior anchors. DogFight [24] is an accurate method for small aerial detection in the dynamic background and performs state-of-the-art performance in two public datasets. However, these methods require a high computational

cost [22], and the efficiency has improved room [5, 24]. Moreover, Dogfight achieves an inference speed of 1 fps on the NVIDIA A6000 GPU, making it unsuitable for deployment on low-performance devices. Additionally, complicated top-bottom detection is not explored in these methods [5, 23].

2.2. *Attention mechanism*

The attention mechanism plays a crucial role for animals in searching for food, avoiding predators, and selecting mates by focusing computational resources on specific regions of the visual field. For example, bumblebees can select flowers of certain colors while ignoring differently colored distractors during visual searches [25]. At the same time, drosophila fixates on the most salient target in a swarm of prey and conspecifics that display different contrasts against complex backgrounds [26]. Additionally, fiddler crabs adjust their escape behavior to minimize combined risk when faced with multiple threats and suppress neural responses to less dangerous predators [27]. Inspired by this characteristic from animals, many research works achieve advanced performance in object detection with a few increasing computational costs. SAG-YOLOv5s [28] is proposed for aerial object detection by introducing the SimAM attention mechanism [29] on YOLOv5s to improve the detection accuracy while reducing the total number of model parameters. Wang *et al.* [30] propose a method based on YOLOX [31] by adding the Squeeze-and-Extraction (SE) module [32] to address the issue of low detection accuracy and reduce the computing resource consumption. VAMYOLOX [33] is further proposed to improve the efficiency of the method [30] by replacing the SE module [32] with Triplet Attention Module (TAM) [34]. To improve the detection accuracy of salient objects, Qin *et al.* [35] develop an attentional dense atrous (dilated) spatial pyramid pooling (AD-ASPP) module to selectively use the local saliency cues captured by dilated convolutions with a small rate and the global saliency cues captured by dilated convolutions with a large rate. However, this dataset [36, 37] is relatively bigger than the aerial object in size. Additionally, the attention mechanism is not always efficient when the plugging position is unsuitable in the different networks for small target detection against complex natural backgrounds. The effectiveness of attention mechanisms requires empirical validation through practical experimentation.

2.3. *Feature fusion*

In modern network architectures, feature fusion is a pervasive technique that combines features from different layers or branches. This process is frequently accomplished through elementary operations such as summation or concatenation. For small object detection, the quality of feature fusion significantly impacts the detection accuracy. Therefore, many researchers have extensively explored this technique. D-A-FS-SSD [38] (Dilated-Attention-Feature Fusion SSD) detects small objects by connecting the dilated convolutional layer and fusing the low-level feature map responsible for detecting small objects with the high-level feature map. Liu *et al.* [39] propose DBF-YOLO based on YOLOv5 detect small objects by fusing shallow features. Zhang *et al.* [40] revisit feature fusion for mining intrinsic RGB-T saliency patterns and propose a novel deep feature fusion network, which consists of the multi-scale, multi-modality, and multi-level feature fusion modules to improve the detection accuracy. However, most feature fusion methods concentrate on enhancing accuracy but neglect the efficiency of small object detection.

Based on the analysis above, additional experimental investigations are necessary to ascertain the effectiveness of attention mechanisms in target detection. The proposition of a feature fusion approach that enhances precision and inference speed is crucial for driving advancements in object detection. The present work proposes a lightweight and effective approach, where we first utilize spatial attention mechanisms for target localization. We employ optimized pooling modules and feature fusion modules to balance speed and accuracy. Ultimately, we achieve a model size of only 4.5 M. Additionally, our approach achieves state-of-the-art performance on five publicly available datasets.

3. Aerial Moving Object Detection Pipeline

3.1. *Framework overview*

Our accurate and energy-efficient network is shown in Fig. 2. To obtain feature maps, we visualize the floating-point values of network layers converted into pixel values and save them as a text file. Subsequently, each text file's pixel values are transformed into images, yielding the final desired feature maps. In the feature extraction stage, we augment the image dataset with a mosaic augmentation to enhance the network robustness during training. We utilize lightweight CSP-Darknet53 (i.e. with a few channels) incorporated Spatial attention mechanism to extract features. The spatial attention and convolutional features are extracted using a residual-spatial attention network (i.e. the yellow block). Compared to the first-level extraction of peripheral contour features (i.e. feature map from layer 1), the attention layer further extracts more profound internal features and more apparent contour characteristics (i.e. feature map from attention layer). This feature map is input into the convolutional downsampling operation to obtain

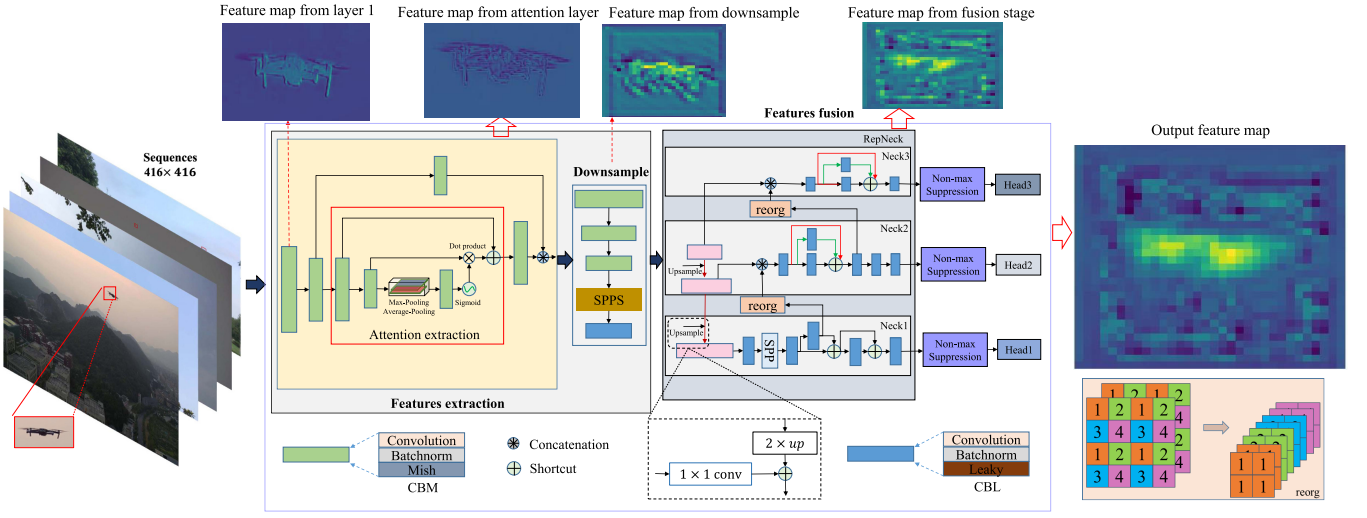
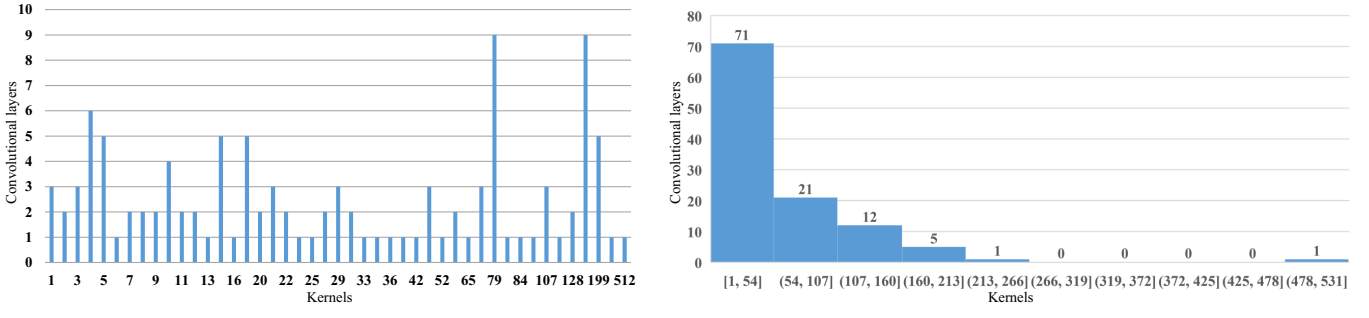


Fig. 2. (Color online) **LENet Framework:** In feature extraction, a set of images is captured by a camera input clip for our network. We utilize a residual network comprising a spatial attention module to extract clear internal features. While filtering out the noise and then carrying out the downsample operation to extract the more depth feature, in which the optimized *SPPS* can provide rich semantic features. In the feature fusion stage, the designed *RepNeck* comprising an upsampling operation (shown in pink color) and a downsampling operation is utilized to transfer more semantic information into the latter layers. Finally, the output of the feature vectors of the *RepNeck* is sent to the detection head and nonmaximum suppression module to obtain the final detection results.

the depth feature. In the feature fusion stage, we first carry out the upsample operations (i.e. the pink block) and add the former layers with the same channels. Downsampling operations are carried out to reduce the parameters and speed up the inference speed, where we utilize reorg layer [41] to downsample among each downsampling operation instead of the convolutional layer to maintain more semantic information and transfer into latter layers, the downsampled feature is concatenated with former upsampled layers to obtain the rich semantic feature information (i.e. feature map from fusion stage). The final result feature map (i.e. output feature map) obtained after applying nonmaximum suppression indicates that this method pays attention to the crucial feature locations of the detected objects. All operations are in an end-to-end network.

The network consists of 180 layers, the most computationally intensive being the 111 convolutional layers. The convolutional layers have varying kernels, ranging from 1 to 512. Figure 3 presents statistics on the number of kernels in the computationally intensive convolutional layers. In Fig. 3(a), the number of kernels is mapped to the corresponding number of convolutional layers, while Fig. 3(b) provides a statistical distribution of kernel numbers. From these two figures, it can be observed that convolutional layers with 79 and 154 kernels have the highest count, with nine convolutional layers. Most convolutional layers have kernel numbers within the range of 1–54 from Fig. 3(b), with 71 convolutional layers. This also implies that more than half of the convolutional layers in the overall network architecture utilize a small number of convolutional



(a) (b)

Fig. 3. Statistics on the number of kernels in computationally intensive convolutional layers. (a) The number of convolutional layers corresponds to the number of kernels. (b) Statistical distribution of kernel numbers.

Table 1. Computational requirement of the LENet.

Component	Feature extraction	Feature fusion
BFLOPs	5.52	2.56

kernels. This design choice helps to reduce parameter count, further decreasing the computational burden and accelerating the inference speed of the model. From Table 1, we determined that the feature extraction stage consumes only 5.52 BFLOPs. In comparison, the computational consumption during the feature fusion stage is 2.56 BFLOPs, demonstrating the low computational costs of our method and indicating its lightweight nature.

3.2. *Spatial features combination with convolutional features*

Human perception relies heavily on attention. Rather than attempting to process an entire scene simultaneously, the human visual system utilizes partial glimpses and selectively focuses on significant elements to capture visual structure better. Inspired by this idea, we obtain the spatial features by refining the feature maps and exploring the inter-spatial relationship of features. The process for computing spatial attention involves applying average-pooling and max-pooling operations along the channel axis for convolutional features, which are then concatenated to create a feature descriptor. This attention method has been demonstrated to highlight informative regions [42] effectively. After concatenating the feature descriptor, a convolution layer is applied to produce a spatial attention map, denoted as $\mathbf{F}_s \in \mathbf{R}^{\mathbf{H}\times\mathbf{W}}$, which indicates where to emphasize or suppress.

After four convolutional operations for the input images, to aggregate channel information from a feature map, we utilize two pooling operations that generate two 2D maps: $\mathbf{F_{savg}} \in \mathbf{R}^{\mathbf{1}\times\mathbf{H}\times\mathbf{W}}$ and $\mathbf{F_{smax}} \in \mathbf{R}^{\mathbf{1}\times\mathbf{H}\times\mathbf{W}}$. These maps represent the average-pooled and max-pooled features across the channel, respectively. They are concatenated and then convolved by a standard convolution layer to produce the 2D spatial attention map. The attention map is added with a convolutional feature map. In summary, the computational process of spatial attention can be formulated as follows:

$$\begin{aligned}
\mathbf{F_1} &= \mathrm{Conv}^{1\times1}(\mathbf{F}),\\
\mathbf{F_2} &= \mathrm{Conv}^{3\times3}(\mathbf{F_1}),\\
\mathbf{F^s_{map}} &= \sigma(\mathrm{Conv}^{7\times7}([\mathrm{AvgPool}(\mathbf{F_2}); \mathrm{MaxPool}(\mathbf{F_2})]))\\
&= \sigma(\mathrm{Conv}^{7\times7}[\mathbf{F_{savg}}; \mathbf{F_{smax}}]),\\
\mathbf{F^r_{att}} &= \mathbf{F} \otimes \mathbf{F^s_{map}},\\
\mathbf{F^s_{att}} &= \mathbf{F^r_{att}} + \mathbf{F},
\end{aligned} \tag{1}$$

where σ represents the sigmoid function and $\mathrm{Conv}^{7\times7}$ denotes a convolutional operation with convolutional kernel of 7×7, the $\otimes$ denotes elements-wise multiplication, F^s_{att} is the final output of spatial attention operation, and $\mathbf{F} \in \mathbb{R}^{C\times H\times W}$ is an intermediate feature map.

The attention feature map is convoluted with a convolutional layer. Followed by a convolutional layer, the obtained feature concatenated with the second convolutional layer will be regarded as the input feature of the downsample stage.

3.3. *SPPS for riching feature*

We argue that the SPP [43] operation for extracting features is complex, influencing inference efficiency. As such, we simplify the extraction feature process with MaxPool of 7×7 to replace the original pool operation and integrate the module into the downsample stage. In this way, more features will be retained, and the same operation extension improves efficiency. Concretely, we define the inputting feature as $\mathbf{F} \in \mathbb{R}^{C\times H\times W}$, the process operation is formulated as follows:

$$\begin{aligned}
\mathbb{F} &= \mathrm{Conv}(\mathbb{F}),\\
\mathbf{F_s} &= \mathrm{Concate}[\mathbf{F_c}, \mathrm{MaxPool}_{7\times7}(\mathbf{F_c}),\\
&\qquad \mathrm{MaxPool}_{7\times7}(\mathbf{F_c}), \mathrm{MaxPool}_{7\times7}(\mathbf{F_c})],\\
\mathbb{F}_{\mathrm{spps}} &= \mathrm{Conv}(\mathbb{F}_s),
\end{aligned} \tag{2}$$

where $\mathbf{F_{spps}} \in \mathbb{R}^{4C\times H\times W}$ denotes the final output through *SPPS*, and the Concate represents the concatenation operation along the channel dimension.

3.4. *Feature fusion with RepNeck*

We make statistical data for the aerial object size in NPS-Drone [44], the small aerial object size distributes in NPS-Drone distribution from 10×5 to 70×20, and all the object sizes with width and height less than 5% of the image size. Recognizing these objects at vastly different scales is a fundamental challenge in computer vision since the object moves against cluttered environments, and using a single high-level feature for detecting small objects is suboptimal. The top-down architecture, which includes lateral connections for constructing high-level semantic feature maps at all scales is an effective design. By augmenting the bottom-up path, accurate localization signals are incorporated into lower layers of the feature hierarchy, which enhances the entire feature hierarchy and shortens the information path between lower layers and the topmost feature. However, the bottom-up path method downsamples by convoluting and missing the semantic information. As such, we design a feature fusion method inspired by ResNet [6] and RepVGG [45] that usually are utilized to

extract features. Unlike ResNet and RepVGG, which perform addition operations between a single convolutional feature layer and the original feature layer, our RepNeck incorporates multiple convolutional layers with the original feature layer through addition. Additionally, we employ the Reorg operation instead of convolutional downsampling. Compared to convolutional downsampling, the Reorg operation does not consume computational resources too much and assures overall computational savings in our method. *RepNeck* shown in Fig. 4 comprises three branch necks for enhancing accuracy and efficiency. We orderly denote three necks as *Neck1*, *Neck2*, and *Neck3* from left to right. The details are introduced as follows.

3.4.1. *Neck1*

After feature extraction, we upsample the feature map three times from top to bottom. Each upsampling operation increases the resolution by a factor of two compared to the previous level. We utilize five convolutional layers to extract features between the bottom upsample and the SPP module. Concretely, given the bottom upsampling feature map $\mathbf{F} \in \mathbb{R}^{C\times H\times W}$, $\mathbf{F_s} \in \mathbb{R}^{C1\times H\times W}$ denotes the outputting of the feature map through five convolutional layers and an SPP module, $\mathbf{F_{r1}} \in \mathbb{R}^{C2\times H\times W}$ denotes the outputting of the feature map from the first residual network, and $\mathbf{F_{r2}} \in \mathbb{R}^{C3\times H\times W}$ denotes the outputting of the feature map from the second residual network, where the C, $C1$, and $C2$ indicate the channels of filters. The H and W represent the height and width of the bottom-level upsampling feature map, respectively. The process of *Neck1* is formulated as follows:

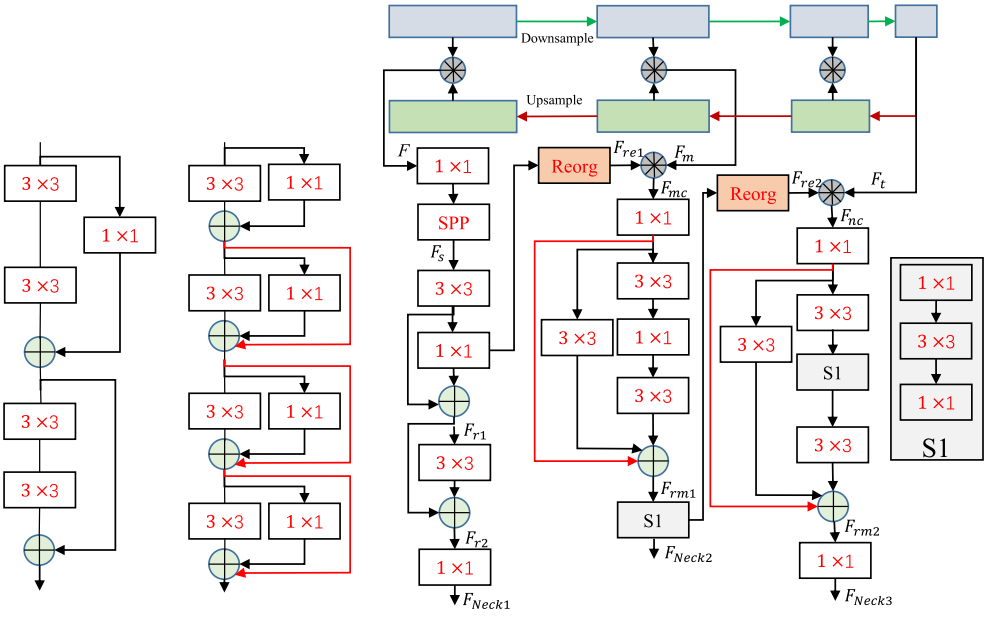
$$\begin{aligned} \mathbf{F_s} &= SPP(\text{Conv1}(\mathbf{F})), \\ \mathbf{F_{r1}} &= \text{Conv2}(\mathbf{F_s}) + \text{Conv}(\mathbf{F_s}), \\ \mathbf{F_{r2}} &= \mathbf{F_{r1}} + \text{Conv}(\mathbf{F_{r1}}), \\ \mathbf{F_{Neck1}} &= \text{Conv}(\mathbf{F_{r2}}), \end{aligned} \tag{3}$$

where Conv1, and Conv2 denote the operations performed after passing through 1 and 2 convolutional layers, respectively. $\mathbf{F_{Neck1}} \in \mathbb{R}^{C4\times H\times W}$ is the final outputting of feature map in *Neck1*, where $C4 = 3 \times (N + 5)$, and N represents the number of object categories detected.

3.4.2. *Neck2*

The intact feature information $\mathbf{F_{re1}} \in \mathbb{R}^{\mathbf{C5}\times\frac{\mathbf{H}}{\mathbf{2}}\times\frac{\mathbf{W}}{\mathbf{2}}}$ is transferred from *Neck1* by downsampling *reorg* operation colored in orange block instead of convolution. $\mathbf{F_{re1}}$ is concatenated with middle upsampling feature map $\mathbf{F_m} \in \mathbb{R}^{C6\times\frac{H}{2}\times\frac{W}{2}}$ to obtained $\mathbf{F_{mc}} \in \mathbb{R}^{(\mathbf{C5}+\mathbf{C6})\times\frac{\mathbf{H}}{\mathbf{2}}\times\frac{\mathbf{W}}{\mathbf{2}}}$, and then followed by a parallel branch inspired by RepVGG [45]. The operation process is formulated as follows:

$$\begin{aligned} \mathbf{F_{mc}} &= \text{Concate}[\mathbf{F_m}, \mathbf{F_{re1}}], \\ \mathbf{F_{rm1}} &= \text{Conv2}(\mathbf{F_{mc}}) + \text{Conv4}(\mathbf{F_{mc}}) + \text{Conv}(\mathbf{F_{mc}}), \\ \mathbf{F_{Neck2}} &= \text{Conv3}(\mathbf{F_{rm1}}), \end{aligned} \tag{4}$$



(a) ResNet (b) RepVGG (c) RepNeck

Fig. 4. Sketch of RepNeck architecture. RepNeck has three subpart *Neck1*, *Neck2*, and *Neck3*. Inspired by ResNet [6] and RepVGG [45], we employed a binary and ternary branching structure.

where $\mathbf{F_{rm1}} \in \mathbb{R}^{C7\times\frac{H}{2}\times\frac{W}{2}}$ is the output feature map of parallel branch network. $\mathbf{F_{Neck2}} \in \mathbb{R}^{C4\times\frac{H}{2}\times\frac{W}{2}}$ is the final output feature map of *Neck2*.

3.4.3. *Neck3*

Similar with *Neck2*, we denote the $\mathbf{F_t} \in \mathbb{R}^{C7\times\frac{H}{4}\times\frac{W}{4}}$ as the top feature map before upsampling operation, $\mathbf{F_{re2}} \in \mathbb{R}^{C8\times\frac{H}{4}\times\frac{W}{4}}$ is transformed from *Neck2*, $\mathbf{F_{nc}} \in \mathbb{R}^{(C7+C8)\times\frac{H}{4}\times\frac{W}{4}}$ is the concatenation feature map from $\mathbf{F_t}$ and $\mathbf{F_{re2}}$ with the operation of *reorg*. The operation process is formulated as follows:

$$\begin{aligned}\mathbf{F_{nc}} &= \text{Concate}[\mathbf{F_t},\ \mathbf{F_{re2}}],\\ \mathbf{F_{rm2}} &= \text{Conv2}(\mathbf{F_{nc}}) + \text{Conv6}(\text{Conv}(\mathbf{F_{nc}}))\\ &\quad + \text{Conv}(\mathbf{F_{nc}}),\\ \mathbf{F_{Neck3}} &= \text{Conv}(\mathbf{F_{rm1}}),\end{aligned} \tag{5}$$

where the $\mathbf{F_{Neck3}} \in \mathbb{R}^{C4\times\frac{H}{4}\times\frac{W}{4}}$ is the final output of *Neck3*.

3.5. *Loss function*

We implement three loss functions to optimize our method: (i) Objectness loss. (ii) Classification class. (iii) Localization loss.

3.5.1. *Objectness loss*

The feature map F_{map} is considered as an $S \times S$ grid, and B bounding boxes are predicted in each cell, and the prediction losses are applied. As shown in the following equation, the objectness loss is calculated by following the condition if the object is present or not in the cell.

$$\begin{aligned}L_{\text{obj}} &= \sum_{m=0}^{S^2}\sum_{n=0}^{B}\mathbb{A}_{mn}^{\text{obj}}(c_m - \hat{c_m})^2\\ &= \lambda_{\text{noobj}}\sum_{m=0}^{S^2}\sum_{n=0}^{B}(1-\mathbb{A}_{mn}^{\text{obj}})(c_m - \hat{c_m})^2,\end{aligned} \tag{6}$$

where $\mathbb{A}_{mn}^{\text{obj}}$ is indicator binary function which takes value 1 if nth bounding box in cell m contains the object. λ_{noobj} is a hyperparameter that is set to 5. c_m and $\hat{c_m}$ denote the predicted and ground-truth confidence scores, respectively.

3.5.2. *Classification loss*

As shown in Eq. (7), a class-specific loss is computed using binary cross entropy:

$$\begin{aligned}L_{\text{cls}} = \sum_{m=0}^{S^2}\sum_{n=0}^{B}\mathbb{A}_{m,n}^{\text{obj}}\sum_{c\in\text{classes}}&\hat{p_n}(c)\log(p_n(c)\\ &+ (1-\hat{p_n}(c))\log(1-p_n(c))),\end{aligned} \tag{7}$$

where $p_i(c)$ and $\hat{p_i}(c)$ denote the probability scores from the predicted class and ground-truth class.

3.5.3. *Localization loss*

The localization loss can be defined as follows:

$$\begin{aligned}L_{\text{loc}} &= 1 - \text{IOU} + \frac{\rho^2(b,\hat{b})}{c^2} + \beta\phi,\\ \text{IOU} &= \frac{|B\cap\hat{B}|}{|B\cup\hat{B}|},\quad \beta = \frac{v}{(1-\text{IOU})+v},\\ \phi &= \frac{4}{\pi^2}\left(\arctan\frac{\hat{\omega}}{\hat{h}} - \arctan\frac{w}{h}\right)^2,\end{aligned} \tag{8}$$

where $\hat{B} = (\hat{x},\hat{y},\hat{\omega},\hat{h})$ denotes the ground-truth box, and $B = (x,y,\omega,h)$ denotes the predicted box. b and $\hat{b}$ represent the central points of B and $\hat{B}$, $\rho(\cdot)$ is the Euclidean distance, and c denotes the diagonal length of the smallest enclosing box covering the two boxes. The total loss is formulated as follows:

$$\text{Loss}_{\text{total}} = \lambda_1 L_{\text{obj}} + \lambda_2 L_{\text{cls}} + \lambda_3 L_{\text{loc}}, \tag{9}$$

where $\lambda_1 = 1, \lambda_2 = 1,$ and $\lambda_3 = 0.07$.

4. Experiment

4.1. *Implementation details*

4.1.1. *Training phase*

We train our LENet on the MFNet [21], Det-Fly [46], TIB-Net [5], NPS-Drone [44], and DUT [47], respectively. When training the model on the five datasets by using 1 GTX 3090 GPU, we choose the stochastic gradient descent optimizer (SGD) with an initial learning rate $\eta_{\text{initial}} = 1.3 \times 10^{-3}$. At 80% and 90% of the setting iterations, the learning rate drops to 10 times the previous learning rate. The weight decay is 0.0005, and the momentum is 0.949, respectively. The other parameters are shown in Table 2.

4.1.2. *Test phase*

In Det-Fly [46], we test the inference time and BFLOPs on RTX 3090 GPU. The testing of the remaining datasets follows the principles outlined in the original paper. In each testing experiment, for testing inference time and BFLOPs. We input the same original resolution image, and the accuracy (mAP) is gauged on the testing set.

4.2. *Compared methods results*

MFNet. The detection performance of the proposed method, as well as two representative detectors, namely

Table 2. Computational requirement of the LENet.

Dataset	Input-size	Iteration	Batchsize	Mini-Batchsize
MFNet [21]	[416, 416]	200 k	64	4
Det-Fly [46]	[416, 416]			
	[640, 640]	40 k	64	32
	[1024, 1024]			
TIB-Net [5]	[1024, 1024]	200 k	64	32
NPS-Drone [44]	[1024, 1024]	200 k	64	32
DUT [47]	[640, 640]	200 k	64	8

YOLOv5s [8] and MFNet [21], is presented in Table 3. Our method demonstrates a significant improvement over the state-of-the-art approaches. Concretely, our method exhibits several advantages, including lower computational costs measured in BFLOPs, fewer parameters, and higher Intersection over Union (IOU) scores. For instance, compared to the well-performing MFNet-M, our model is more than half its size while achieving a nearly three-point improvement in detection accuracy. Notably, our model outperforms MFNet-M by approximately 17 points regarding IOU, yet our computational resource consumption is only about one-ninth of MFNet-M. These findings highlight that despite the compact size of our LENet, it maintains high accuracy and localization quality. Furthermore, the qualitative visualization in Fig. 5 vividly illustrates the superiority of our method compared to the advanced MFNet, particularly in challenging background scenarios. Notably, the samples in the first three rows of Fig. 5 were captured from a distance. Our method consistently achieves superior detection performance, with an average confidence score of 94.8 compared to 87.0 for MFNet. Furthermore, from these qualitative visualization results, it can be observed that our method demonstrates excellent performance not only in detecting small objects but also in detecting large objects. Our method outperforms MFNet and YOLOv5s in multi-object and single-object detection tasks.

TIB-Net. We present detailed comparisons of our method with other state-of-the-art techniques in Table 4. Although our approach possesses a bigger model size than the advanced TIBNet [5], we achieve higher accuracy and low latency by a large margin. This is due to the extensive usage of attention layers in TIBNet [5] that can slow the inference speed. This is also reflected in our ablation study in Sec. 4.3, where we observed the impact of attention layers on inference speed. Furthermore, our method outperforms some approaches based on heavyweight backbones, such as Faster RCNN [48] (based on ResNet50).

Det-Fly. The experimental results on the Det-Fly [46] dataset are displayed in Table 5. Our method performs state-of-the-art in terms of both mAP and latency. Moreover, the model size of our method is 3% of the advanced method YOLOv7 [10], and the memory footprint is lower than YOLOv7 [10], which means our method enjoys a low computational complexity. Compared to the state-of-the-art model YOLOv7X, our model achieves a two-point higher accuracy while possessing faster inference speed, smaller model size, and lower GPU memory usage and computational resource consumption than YOLOv7X. As shown in Fig. 6, we visualize the three methods' mAP, latency, and BFLOPs. Our red curve encloses the other two, demonstrating our method has a relative trade-off among accuracy, latency, and BFLOPs. Also, our method achieves superior accuracy and inference speed with minimal computational resource consumption, which is favorable for devices with weaker computational capabilities. Such performance is competitive for deploying it on edge-computing devices. The qualitative visualization shown in Fig. 7 also exhibits the effectiveness of LENet compared to YOLOv7 and YOLOv7X. Although all methods detect the target, we possess a higher detection score (98.5 vs 76.6 on average).

NPS-Drone. Table 6 presents the quantitative benchmarking results on NPS-Drones [44], where our LENet outperforms all other approaches in terms of mAP. Although LENet possesses the lower precision and *F*1-Score than Dogfight [24], LENet achieves higher precision (0.91 vs. 0.89), and recall (0.92 vs. 0.91) than the advanced Dogfight [24]. The improvement in recall suggests that LENet has further potential for enhancement. Upon analysis, the small aerial object size distributes in

Table 3. Quantive benchmarking results on MFNet.

Method	Model size (M)↓	FLOPs (B)↓	mAP (%)↑	IOU (%)↑
YOLOv5s [8]	14.9	—	90.6	49.4
MFNet-S [21]	5.2	18.9	90.8	49.1
MFNet-M [21]	9.9	75.3	91.5	51.1
MFNet-L [21]	19.5	157.6	90.9	49.0
LENet (ours)	**4.5**	**8.1**	**94.3**	**67.8**

Notes: The best method is shown in Red and the second best method is shown in Blue. ↑ (↓) indicates that larger (smaller) values lead to better (worse) performance.

Fig. 5. Qualitative visualization on comparison with state-of-the-art methods. Each row represents the different scenarios. Each column demonstrates the visualization results from YOLOv5s, MFNet-S, MFNet-M, MFNet-L, and LENet from left to right. LENet has higher confidence scores than MFNet, which can demonstrate accurate detection performance. Please zoom in for the best view.

Table 4. Quantive benchmarking results on TIBNet.

Method	Backbone	mAP (%)↑	Latency (ms)↓	Model size↓
Faster RCNN [48]	ResNet50	87.2	217	333.5 M
Faster RCNN [48]	MobileNet	67.5	125	162.5 M
Casecade RCNN [7]	MobileNet	78.0	164	384.9 M
YOLOv3 [49]	DarkNet53	84.9	66	234.1 M
YOLOv4 [50]	CSPDarkNet53	86.0	19	256.0 M
YOLOv5 [8]	YOLOv5s	86.2	5	14.9 M
EXTD [51]	MobileFaceNet	85.1	274	696.9 KB
TIBNet [5]	IB-Net	89.2	290	697.0 KB
DTD-V4-Tiny [23]	—	85.1	—	1.4 M
LENet (ours)	—	**90.0**	**26**	**4.5 M**

Table 5. Quantitative benchmarking results on det-fly.

Method	Image size	mAP (%) ↑	Latency (ms)↓	Model size (M) ↓	Memory footprint (M)↓	FLOPs (B)↓
YOLOv7 [10]	[416,416]	83.3	6.9	139.4	1607	43.6
	[640,640]	89.9	11.0	139.4	2179	103.2
	[1024,1024]	93.8	23.0	139.4	3447	264.3
YOLOv7X [10]	[416,416]	86.1	14.9	270.4	1995	79.5
	[640,640]	90.1	15.8	270.4	2603	188.1
	[1024,1024]	94.2	33.5	270.4	4387	481.4
LENet (ours)	[416,416]	91.6	4.3	4.5	1135	8.1
	[640,640]	95.0	6.9	4.5	1441	19.1
	[1024,1024]	96.3	13.6	4.5	2209	49.0

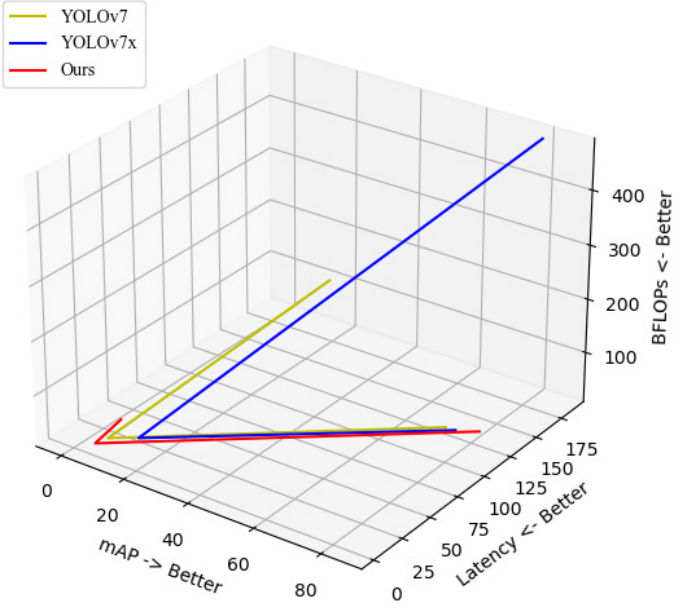


Fig. 6. Trade-off performance of accuracy, latency, BFLOPs on Det-Fly. This left arrow indicates that the closer to the left, the better, and vice versa.

NPS-Drone [44] distribution from 10 × 5 to 70 × 20, and all the object sizes with width and height less than 5% of the image size. These objects in NPS-Drone tend to occupy a small portion of the field of view in real-world settings. Accordingly, most common detection methods may not be suitable for detecting such small targets.

DUT. On the DUT [47] dataset, our LENet achieves similar performance with state-of-the-art methods yet processes many frames simultaneously. As shown in Table 7, the original paper's inference speed was tested on a GTX 2080 super. Due to the unavailability of this device model (GTX 2080 super) at our disposal, we use an RTX 3060 that is inferior to the original paper of RTX 2080 super to test the latency. LENet still gets better accuracy and faster speed than other state-of-the-art methods by a large margin on DUT [47]. Specifically, compared to the highest-accuracy model Cascade-RCNN, our model achieves a 0.8-point higher accuracy while being 8× faster in inference speed. Compared to the fastest model YOLOX, our model remains faster while achieving an almost 30-point higher detection accuracy. Such performance is highly favorable and competitive for deploying our method on low-performance computing devices.

Upon analysis, our proposed method obtains the state-of-the-art performance on the five datasets compared with all the baseline methods. Figure 8 shows the representative small object images and their corresponding visualization results in TIBNet [5], NPS-Drone [44], and DUT [47]. Our model demonstrates excellent performance in detecting small targets with high confidence in different scenarios. Specifically, the third row in Fig. 7 and the second row in Fig. 8 represent top-to-bottom detection scenarios with complex backgrounds. Despite the challenging background conditions, our model still performs well. This indicates that our model performs well in detecting aerial targets at long distances, making it highly competitive for applications such as object video surveillance and warning systems, particularly for airborne intruder drones.

4.3. *Ablation study*

To validate the contributions of aerial object detection and scale estimation to detection improvement, we conduct extensive experiments on Det-Fly [46]. In the following experiments, the input size of the detector in the test phase is set to 1024 × 1024 pixels. To validate if the module can consistently improve performance by removing or replacing related modules.

Effect of RepNeck. We use a neck without multiple branches and apply a convolution layer for downsampling (i.e. remove the red line, skip connection with 3 × 3 convolutional layer, and reorg in Fig. 4). The experimental results are listed in Table 8. We note that the mAP, model size, and FLOPs decrease. Inversely, the latency increases without *RepNeck*, which means the inclusion of *RepNeck*

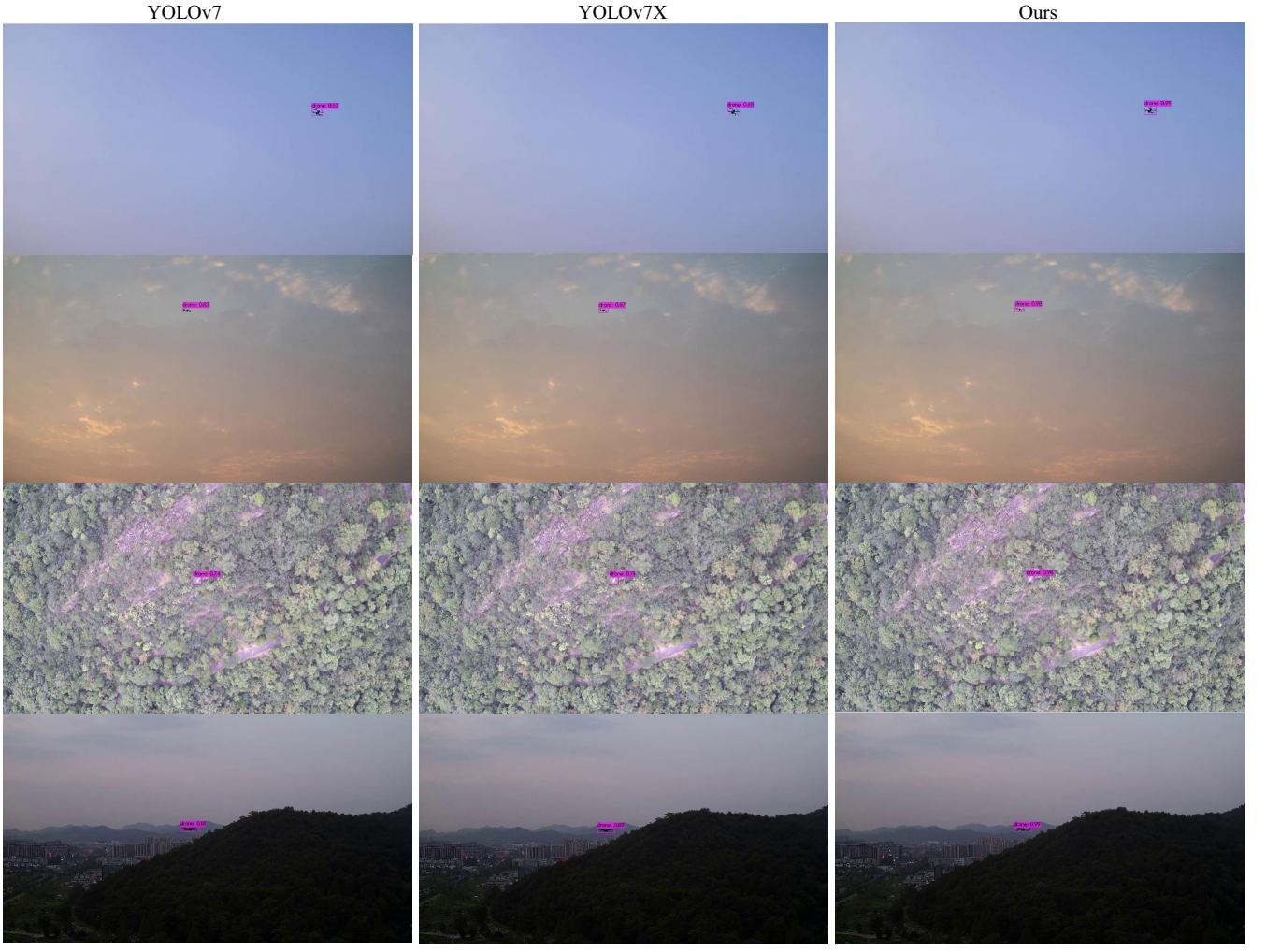


Fig. 7. Qualitative visualization on comparison with state-of-the-art methods. Each row represents the different scenarios. Each column demonstrates the visualization from YOLOv7, YOLOv7X, and LENet from left to right. Please zoom in for the best view.

Table 6. Quantitative benchmarking results on NPS-drones.

Method	Precision (%)↑	Recall (%)↑	F1 Score (%)↑	mAP (%)↑
SCRDet-H [52]	81	74	77	65
SCRDet-R [52]	79	71	75	61
FCOS [53]	88	84	86	83
Mask-RCNN [54]	66	91	76	89
MEGA [55]	88	82	85	83
SLSA [8]	47	67	55	46
Dogfight [24]	92	91	92	89
LENet (Ours)	88	92	90	91

leads to improvements in accuracy and latency while also introducing a high computational cost. This is because the *RepNeck* has introduced many Reorg operations while operating maintains more feature information. The Reorg operation in *RepNeck* is simply a way of recombining information instead of convolution. As such, the latency improves, and model size increases if adding *RepNeck*. For RepNeck, the trade-off of a small increase in parameter

Table 7. Quantitative benchmarking results on DUT.

Method	Backbone	mAP (%)↑	Latency (ms)↓
Faster-RCNN [48]	ResNet50	65.3	78.1
	ResNet18	60.5	51.5
	VGG16	63.3	107.5
Casecade-RCNN [7]	ResNet50	68.3	93.5
	ResNet18	65.2	68.0
	VGG16	66.7	125.0
ATSS [56]	ResNet50	64.2	75.2
	ResNet18	61.0	48.8
	VGG16	64.1	105.3
YOLOX [31]	ResNet50	42.7	46.1
	ResNet18	40.0	18.6
	VGG16	55.1	43.5
	Darknet	55.2	19.5
SSD [57]	VGG16	63.2	30.1
LENet (Ours)	SCANet	69.1	11.9

count has proven worthwhile as it has improved accuracy and speed.

Effect of Attention. From Table 8, since the pooling operation does not introduce any extra parameters except for one convolutional layer in attention, and the operation can filter out noisy feature information, we note that the latency decreases by a large. However, the model size and FLOPs do not change much compared with LENet without attention, meaning attention operation enhances the accuracy but degrades the efficiency.

As shown in Fig. 9, we present the color feature maps with and without integrated attention mechanisms (first two rows). For the convenience of visual observation, we also display grayscale feature maps (the last two rows). The first and third rows represent the feature maps without an attention mechanism, while the second and fourth rows represent those with an attention mechanism. The first column corresponds to the input image, and the subsequent three columns represent outputs from different layers. From the second column, it can be observed that our method extracts more prominent features and clearer contours compared to the case without an attention mechanism. Similarly, the third column demonstrates that our method produces sharper internal and contour features relative to the absence of an attention mechanism. Additionally, the feature map output from the final layer

Fig. 8. Representative images of the small aerial object image sequences and their corresponding visualization results in [5, 44, 47], in which each image represents a different scenario. Please zoom in for the best view.

Table 8. Ablation study on det-fly.

Ablation setting	mAP (%)↑	Latency (ms)↓	Model size (M)↓	FLOPs (B)↓
w/o *RepNeck*	95.2 (↓1.1)	14.5 (↑0.9)	3.7 (↓0.8)	19.1 (↓21.0)
w/o *Attention*	95.6 (↓0.7)	7.8 (↓5.8)	4.4 (↓0.1)	47.6 (↓1.4)
w/o *SPPS*	95.8 (↓0.5)	13.8 (↑0.2)	4.5	49.0
LENet (Baseline)	**96.3**	**13.6**	**4.5**	**49.0**

Table 9. Our method displays the temperature and power consumption before and after its execution.

Static temperature		Running temperature		Increment		Static power	Running power	Increment
CPU:46.12°C	GPU:43.69°C	CPU:49.03°C	GPU:47.19°C	CPU:2.91°C	CPU:3.50°C	0.582 W	3.50 W	2.918 W

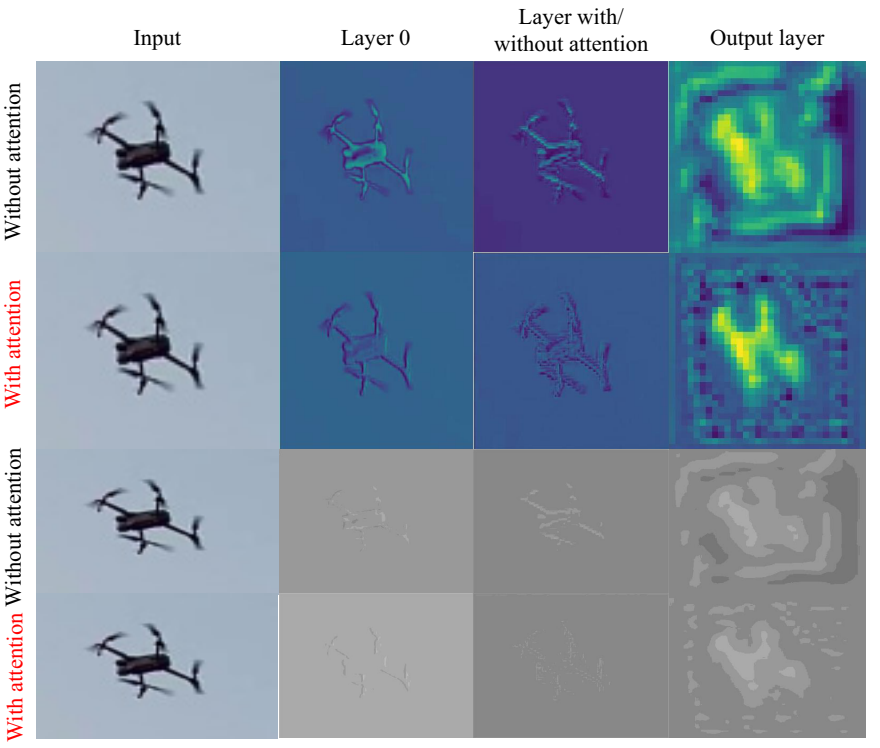


Fig. 9. (Color online) Feature map extracted from different layers. The first column represents the input image, the second column shows the output feature map from the first layer, the third column displays the output feature maps with and without attention mechanism, and the fourth column exhibits the final output feature map. The grayscale feature maps are provided to aid in observing the effects without being influenced by color images.

(last column) reveals that our method pays more attention to relevant object features while ignoring irrelevant information. Figure 10 provides a qualitative visualization of the detection results for both cases. It is evident that the integration of the attention mechanism leads to higher detection accuracy (0.99 vs. 0.98, 0.92 vs. 0.74). The feature map visualization and detection result visualization analysis confirm the effectiveness of integrating the attention mechanism in our approach.

Effect of SPPS. After replacing SPP with SPPS in the downsampling stage, we note that the speed of processed images increases to 13.8 ms. This is because the SPP module partitions the feature map into pieces with different-size pooling operations, aggravating the complexity problem when performing pool operations. While the operation does change the number of parameters, the model size and FLOPs are not improved. In addition, we see that the mAP decreases 0.5% because pooling operations with different sizes can obtain extra noisy feature information.

To further demonstrate the effects of *SPPS* on accuracy and latency, we replace SPP with SPPS in YOLOv3-SPP [49]. As shown in Table 10, compared with YOLOv3, the SPPS improves the accuracy and the latency. Inversely, both accuracy and latency were enhanced compared with YOLOv3-SPP.

4.4. *Deployment for testing: Jetson NVIDIA Orin NX*

To demonstrate the deployment capability of LENet on edge-computing devices, we deploy our LENet on an onboard computer, NVIDIA Jetson Orin NX device with 16 GB GPU memory and 8 CPU cores. To demonstrate whether the temperature remains within the normal operating range

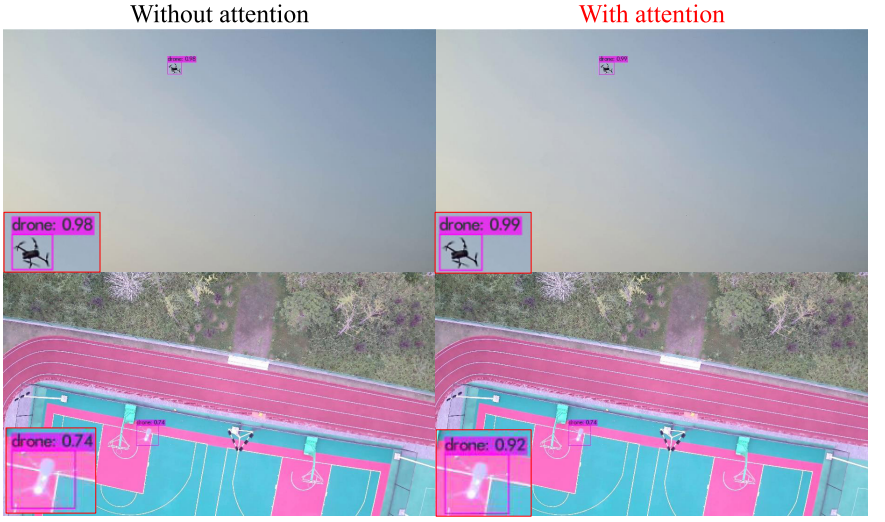


Fig. 10. Qualitative visualization on comparison with state-of-the-art methods. Each row represents the different scenarios. From left to right, each column demonstrates the visualization results without attention and with attention. Please zoom in for the best view.

Table 10. Ablation study of YOLOv3 on det-fly.

Ablation setting	mAP (%)↑	Latency (ms)↓
YOLOv3 [49]	87.8	6.3
YOLOv3-SPP [49]	87.5	6.5
YOLOv3-SPPS	88.4	6.4

(i.e. below 60° Celsius) during the execution of our method, we conduct temperature experiments. With input frames at a resolution of 640 × 480, our 416 resolution model can accurately detect the drones and achieve **14.8 fps** without any optimization and keeping the board temperature well below **50**°C. Compared to the static state, the temperature increase is insignificant when our model runs. Furthermore, the power consumption on the Orin NX platform is only **2.9 W**, confirming that our network satisfies the portability, practicality, and energy-efficient requirements. Table 9 shows the sensors' temperature and power consumption values in two scenarios.

5. Conclusion

In this study, we proposed an energy-efficient and accurate approach for detecting aerial moving objects. The suggested approach is based on attention mechanism and feature fusion. The proposed method's overall efficiency was evaluated using five standard datasets. The performance assessment shows a satisfactory balance between the proposed method's detection accuracy, inference time, energy efficiency, and memory footprint. In these respects, our approach effectively relieves the burden of processing video sequences for resource-constrained surveillance devices. We also further deploy our method on edge-computing devices to test to demonstrate the feasibility and portability of our method. For future work, we plan to extend the method to other moving object domains, such as moving robotics and cars, and improve its detection efficiency. Our research demonstrates the proposed method's effectiveness and practicality in aerial moving object detection, paving the way for its application in various real-world scenarios.

ORCID

Xunkuai Zhou https://orcid.org/0000-0003-4466-8937
Li Li https://orcid.org/0000-0001-5097-9972
Ben M. Chen https://orcid.org/0000-0002-3839-5787

References

[1] K. Feng, W. Li, J. Han and F. Pan, Low-latency aerial images object detection for UAV, *Unmanned Syst.* **10**(1) (2022) 57–67.
[2] D. Seo and J. Kang, Collision-avoided tracking control of uav using velocity-adaptive 3d local path planning, *Int. J. Control Autom. Syst.* **21**(1) (2023) 231–243.
[3] B. M. Chen, On the trends of autonomous unmanned systems research, *Engineering* **12** (2021) 20–23.
[4] A. Coluccia *et al.*, Drone-vs-bird detection challenge at ieee avss2021, in *17th IEEE Int. Conf. Advanced Video and Signal Based Surveillance* (IEEE, 2021), pp. 1–8.
[5] H. Sun, J. Yang, J. Shen, D. Liang, N. Liu and H. Zhou, TIBNet: Drone detection network with tiny iterative backbone, *IEEE Access* **8** (2020) 130697–130707.

[6] K. He, X. Zhang, S. Ren and J. Sun, Deep residual learning for image recognition, in *Proc. IEEE Conf. Computer Vision and Pattern Recognition* (IEEE, 2016) pp. 770–778.

[7] Z. Cai and N. Vasconcelos, Cascade R-CNN: Delving into high quality object detection, in *Proc. IEEE Conf. Computer Vision and Pattern Recognition* (IEEE, 2018), pp. 6154–6162.

[8] G. Jocher, YOLOv5 by Ultralytics (2020), https://github.com/ultralytics/yolov5.

[9] C. Li, L. Li, H. Jiang, K. Weng, Y. Geng, L. Li, Z. Ke, Q. Li, M. Cheng, W. Nie, Y. Li, B. Zhang, Y. Liang, L. Zhou, X. Xu, X. Chu, X. Wei and X. Wei, YOLOv6: A single-stage object detection framework for industrial applications, preprint (2022), arXiv:2209.02976.

[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. Computer Vision and Pattern Recognition* (IEEE, 2023), pp. 7464–7475.

[11] T.-Y. Lin, P. Goyal, R. Girshick, K. He and P. Dollár, Focal loss for dense object detection, in *Proc. IEEE Int. Conf. Computer Vision* (IEEE, 2017), pp. 2980–2988.

[12] 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. Computer Vision* (IEEE, 2021), pp. 2778–2788.

[13] Q. Zhao, B. Liu, S. Lyu, C. Wang and H. Zhang, TPH-YOLOv5++: Boosting object detection on drone-captured scenarios with cross-layer asymmetric transformer, *Remote Sensing* **15**(6) (2023) 1687.

[14] Y. Jiang, Z. Tan, J. Wang, X. Sun, M. Lin and H. Li, GiraffeDet: A heavy-neck paradigm for object detection, preprint (2022), arXiv: 2202.04256.

[15] M. Zhang, F. Lin and B. M. Chen, Vision-based detection and pose estimation for formation of micro aerial vehicles, in *13th Int. Conf. Control Automation Robotics & Vision* (IEEE, 2014), pp. 1473–1478.

[16] J. A. Ramirez-Quintana and M. I. Chacon-Murguia, Self-adaptive SOM-CNN neural system for dynamic object detection in normal and complex scenarios, *Pattern Recogn.* **48**(4) (2015) 1137–1149.

[17] D. Liang, D. Zhang, Q. Wang, Z. Wei and L. Zhang, CrossNet: Cross-scene background subtraction network via 3d optical flow, *IEEE Trans. Multimedia* (2023) 1–14, doi: 10.1109/TMM.2023.3266608.

[18] Z. Zhang, Y. Chang, S. Zhong, L. Yan and X. Zou, Learning dynamic background for weakly supervised moving object detection, *Image Vision Comput.* **121** (2022) 104425.

[19] Z. Chen, R. Wang, Z. Zhang, H. Wang and L. Xu, Background-foreground interaction for moving object detection in dynamic scenes, *Inf. Sci.* **483** (2019) 65–81.

[20] X. Ou, P. Yan, Y. Zhang, B. Tu, G. Zhang, J. Wu and W. Li, Moving object detection method via resnet-18 with encoder-decoder structure in complex scenes, *IEEE Access* **7** (2019) 108152–108160.

[21] M. Dil, M. U. Khan, M. Z. Alam and F. A. Orakazi, Safespace MFNet: Precise and efficient multifeature drone detection network, *IEEE Trans. Vehicle Technol.* **73**(3) (2022) 1–13.

[22] J. Wang, G. Zhang, K. Zhang, Y. Zhao, Q. Wang and X. Li, Detection of small aerial object using random projection feature with region clustering, *IEEE Trans. Cybern.* **52**(5) (2020) 3957–3970.

[23] R. Jiang, Z. Ye, Y. Peng, G. Xie and H. Du, Lightweight target detection algorithm for small and weak drone targets, *Progress Laser Optoelectron.* **59**(8) (2022) 109–120.

[24] M. W. Ashraf, W. Sultani and M. Shah, Dogfight: Detecting drones from drones videos, in *Proc. IEEE/CVF Conf. Computer Vision and Pattern Recognition* (IEEE, 2021), pp. 7067–7076.

[25] V. Nityananda and J. G. Pattrick, Bumblebee visual search for multiple learned target types, *J. Exp. Biol.* **216**(22) (2013) 4154–4160.

[26] P. Sareen, R. Wolf and M. Heisenberg, Attracting the attention of a fly, *Proc. Natl. Acad. Sci.* **108**(17) (2011) 7230–7235.

[27] Z. M. Bagheri, C. G. Donohue and J. M. Hemmi, Evidence of predictive selective attention in fiddler crabs during escape in the natural environment, *J. Exp. Biol.* **223**(21) (2020) jeb234963.

[28] 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* **22**(15) (2022) 5825.

[29] L. Yang, R.-Y. Zhang, L. Li and X. Xie, Simam: A simple, parameter-free attention module for convolutional neural networks, in *Int. Conf. Machine Learning* (PMLR 2021), pp. 11 863–11 874.

[30] C. Wang, L. Meng, Q. Gao, J. Wang, T. Wang, X. Liu, F. Du, L. Wang and E. Wang, A lightweight UAV swarm detection method integrated attention mechanism, *Drones* **7**(1) (2022) 13.

[31] Z. Ge, S. Liu, F. Wang, Z. Li and J. Sun, YOLOX: Exceeding yolo series in 2021, preprint (2021), arXiv:2107.08430.

[32] J. Hu, L. Shen and G. Sun, Squeeze-and-excitation networks, in *Proc. IEEE Conf. Computer Vision and Pattern Recognition* (IEEE, 2018), pp. 7132–7141.

[33] Y. Yang, X. Gao, Y. Wang and S. Song, VAMYOLOX: An accurate and efficient object detection algorithm based on visual attention mechanism for UAV optical sensors, *IEEE Sensors J.* (2022) **23**(11) (2023) 139–155.

[34] D. Misra, T. Nalamada, A. U. Arasanipalai and Q. Hou, Rotate to attend: Convolutional triplet attention module, in *Proc. IEEE/CVF Winter Conf. Applications of Computer Vision* (IEEE, 2021), pp. 3139–3148.

[35] L. Zhu, J. Chen, X. Hu, C.-W. Fu, X. Xu, J. Qin and P.-A. Heng, Aggregating attentional dilated features for salient object detection, *IEEE Trans. Circuits Syst. Video Technol.* **30**(10) (2019) 3358–3371, 2019.

[36] P. Zhu *et al.*, Visdrone-det2018: The vision meets drone object detection in image challenge results, in *Proc. European Conf. Computer Vision Workshops* (ACM, 2018).

[37] G.-S. Xia, X. Bai, J. Ding, Z. Zhu, S. Belongie, J. Luo, M. Datcu, M. Pelillo and L. Zhang, DOTA: A large-scale dataset for object detection in aerial images, in *Proc. IEEE Conf. Computer Vision Pattern Recognition* (IEEE, 2018), pp. 3974–3983.

[38] Y. Liu, Z. Ding, Y. Cao and M. Chang, Multi-scale feature fusion UAV image object detection method based on dilated convolution and attention mechanism, in *8th Int. Conf. Information Technology: IoT and Smart City* 2020, (Association for Computing Machinery, 2021) pp. 125–132.

[39] H. Liu, X. Duan, H. Chen, H. Lou and L. Deng, DBF-YOLO: UAV small targets detection based on shallow feature fusion, *IEEJ Trans. Electrical Electron. Eng.* **18**(4) (2023) 605–612.

[40] Q. Zhang, T. Xiao, N. Huang, D. Zhang and J. Han, Revisiting feature fusion for RGB-*t* salient object detection, *IEEE Trans. Circuits Syst. Video Technol.* **31**(5) (2020) 1804–1818.

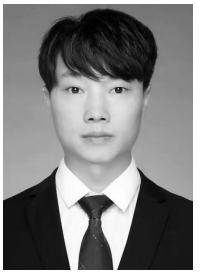
[41] W. Shi, J. Caballero, F. Huszár, J. Totz, A. P. Aitken, R. Bishop, D. Rueckert and Z. Wang, Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network, in *Proc. IEEE Conf. Computer Vision and Pattern Recognition* (IEEE, 2016), pp. 1874–1883.

[42] N. Komodakis and S. Zagoruyko, Paying more attention to attention: Improving the performance of convolutional neural networks via attention transfer, in *Int. Conf. Learning Representations* (OpenReview.net 2017).

[43] K. He, X. Zhang, S. Ren and J. Sun, Spatial pyramid pooling in deep convolutional networks for visual recognition, *IEEE Trans. Pattern Anal. Machine Intell.* **37**(9) (2015) 1904–1916.

[44] 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 *IEEE/RSJ Int. Conf. Intelligent Robots and Systems* (IEEE, 2016), pp. 4992–4997.
[45] X. Ding, X. Zhang, N. Ma, J. Han, G. Ding and J. Sun, RepVGG: Making VGG-style convnets great again, in *Proc. IEEE/CVF Conf. Computer Vision and Pattern Recognition* (IEEE, 2021), pp. 13 733–13 742.
[46] 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 Robotics Autom. Lett.* **6**(2) (2021) 1020–1027.
[47] J. Zhao, J. Zhang, D. Li and D. Wang, Vision-based anti-UAV detection and tracking, *IEEE Trans. Intell. Transp. Syst.* **23** (2022) 25323–25334.
[48] S. Ren, K. He, R. Girshick and J. Sun, Faster RCNN: Towards real-time object detection with region proposal networks, preprint (2016), arXiv:1506.01497.
[49] J. Redmon and A. Farhadi, YOLOv3: An incremental improvement, preprint (2018), arXiv:1804.02767.
[50] A. Bochkovskiy, C.-Y. Wang and H.-Y. M. Liao, YOLOv4: Optimal speed and accuracy of object detection, preprint (2020), arXiv: 2004.10934.
[51] Y. Yoo, D. Han and S. Yun, EXTD: Extremely tiny face detector via iterative filter reuse, preprint (2019), arXiv:1906.06579.
[52] X. Yang, J. Yang, J. Yan, Y. Zhang, T. Zhang, Z. Guo, X. Sun and K. Fu, SCRDet: Towards more robust detection for small, cluttered and rotated objects, in *Proc. IEEE/CVF Int. Conf. Computer Vision* (IEEE, 2019), pp. 8232–8241.
[53] Z. Tian, C. Shen, H. Chen and T. He, FCOS: Fully convolutional one-stage object detection, in *Proc. IEEE/CVF Int. Conf. Computer Vision* (IEEE, 2019), pp. 9627–9636.
[54] K. He, G. Gkioxari, P. Dollár and R. Girshick, Mask *R*-CNN, in *Proc. IEEE Int. Conf. Computer Vision* (IEEE, 2017), pp. 2961–2969.
[55] Y. Chen, Y. Cao, H. Hu and L. Wang, Memory enhanced global-local aggregation for video object detection, in *Proc. IEEE/CVF Conf. Computer Vision and Pattern Recognition* (IEEE, 2020), pp. 10337–10346.
[56] S. Zhang, C. Chi, Y. Yao, Z. Lei and S. Z. Li, Bridging the gap between anchor-based and anchor-free detection via adaptive training sample selection, in *Proc. IEEE/CVF Conf. Computer Vision and Pattern Recognition* (IEEE, 2020), pp. 9759–9768.
[57] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu and A. C. Berg, SSD: Single shot multibox detector, in *Proc. European Conf. Computer Vision* (Springer, Cham, 2016), pp. 21–37.



Xunkuai Zhou is currently pursuing his Ph.D. degree in control science and engineering, at the Tongji University, Shanghai, China. He is currently involved in research as 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 architecture.

Li Li received a Ph.D. degree from the Shenyang Institute of Automation, Chinese Academy of Science, in 2003. She joined Tongji University, Shanghai, China, in 2003, and is now Professor at the Department of Control Science and Engineering.

Her research interests are in data-driven modeling, optimization, and computational intelligence.

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

Dr. Chen is a Fellow of IEEE and 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 an Editor-in-Chief of Unmanned Systems and Editor of the International Journal of Robust and Nonlinear Control.