2024 IEEE International Conference on Robotics and Automation (ICRA 2024)
May 13-17, 2024. Yokohama, Japan

SANet: Small but Accurate Detector for Aerial Flying Object

Xunkuai Zhou[1,2], Benyun Zhao[2], Guidong Yang[2], Jihan Zhang[2], Li Li[1] and Ben M. Chen[2]

***Abstract*— This paper proposes SANet, a small but accurate detector for aerial flying objects. The detector introduces an attention module into the feature extraction module (FEM) for enhancing the accuracy. This FEM with fewer convolutional kernel channels can reduce the parameters, speed up the inference time, and mitigate the computational burden. Furthermore, we optimize the Spatial Pyramid Pooling (SPP) module to enhance both the accuracy and speed. By analyzing the structure characteristic of the *ResNet* and *RepVGG* network that are usually utilized to extract features, a feature fusion module named *RepNeck* is designed to comprehensively fuse features extracted by the FEM, further enhancing the speed and accuracy. Eventually, we develop a neural network with an impressively small model size of only *4.5M*. This network can achieve the state-of-the-art performance on three challenging datasets. Apart from its superior performance, our approach enjoys a real-time detection speed of 14.8 frames per second (fps) and power consumption of only 2.9W while the CPU and GPU temperatures are maintained below $50^{\circ}C$ even on an edge-computing device, highlighting the practicality of our approach for long-duration flying object detection and monitoring tasks.**

I. Introduction

Flying object detection is a crucial research area in machine vision and image processing, which guides the behavior of observers including animals and robots. In complex dynamic environments, autonomous robot systems will need to detect object motion to understand the movement intention, predict the future paths, and react appropriately [1]–[5]. Additionally, detecting potentially dangerous objects early and accurately would provide enough time for autonomous systems such as Unmanned Aerial Vehicles (UAVs) to respond timely [6]–[9], thereby maintaining or reinforcing their autonomous intelligence ability in the interaction and competition. However, the flying object will appear as a dim speckle on the image from the observer's or ontology's insight, it is arduous for the detection task to extract enough visual features [10]. As shown in Fig. 1, these aerial objects with a few pixels are almost invisible to the human eye.

The irregular shapes and sizes of aerial objects are influenced by the generated noise around them from the video codec standard, leading to false or missing detection [11]. The color variation of aerial objects can be significantly affected by dynamic backgrounds, such as moving clouds and illumination changes, causing methods based on color extraction to suffer from a poor detection performance [12], [13]. In particular, aerial detection tasks encompass not only bottom-up but also top-down object detection. In contrast to the former with a singular background such as the sky, the latter faces more complex ground backgrounds and lighting variations, further complicating aerial flying objects detection [14]–[16]. The scale of aerial objects can vary significantly across frames, which presents a significant challenge to detect objects accurately since flying objects have no physical boundaries that constrain their movement in space.

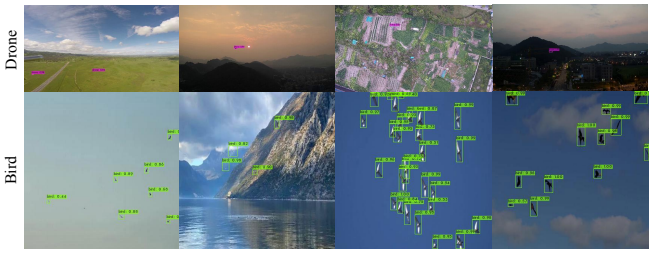


Fig. 1. Qualitative visualization of small moving aerial objects detection in many challenging situations. In each subplot, the drone and bird manifest as faint speckles, occupying just a few pixels, with the majority of their visual attributes challenging to distinguish. Notably, they exhibit exceedingly low contrast against the cluttered backgrounds, amidst variations in illumination

Numerous deep convolutional neural network-based (DCNN) detection methods have followed the scheme of feature extraction, feature fusion, target classification, and localization [17]–[19]. Examples of such methods also include Cascade R-CNN [20], YOLO [21], [22], and RetinaNet [23], which have made significant advancements in general object detection. However, due to the difficulty in learning rich representations from the tiny and poor-quality appearances of small objects, these methods often struggle to detect them in dynamic and cluttered environments. Many researchers have focused on detecting small objects using improved DCNN-based algorithms. Zhu *et al.* [24] propose an improved algorithm based on the YOLOv5 [21], TPH-YOLOv5, for small object detection by replacing the CNN-based prediction heads with a transformer prediction heads (TPH). To reduce the complexity and improve the detection speed of TPH-YOLOv5, they propose the TPH-YOLOv5++ [25] by replacing the prediction head of TPH-YOLOv5 with the cross-layer asymmetric transformer and trained on VisDrone-DET2021 [26]. Jiang *et al.* propose GiraffeDet [27] inspired by DensNet [28] to detect both the small and big objects adaptively. They argue that the main factor affecting detection performance is feature fusion rather than feature extraction. As such, they incorporate more convolutional

The work was supported in part by the Research Grants Council of Hong Kong SAR under Grant 14217922 (*Corresponding author: Benyun Zhao*)

[1]School of Electronics and Information Engineering, Tongji University, Jiading, Shanghai 201804, China. (e-mail: 2010474@tongji.edu.cn; lili@tongji.edu.cn).

[2]Department of Mechanical and Automation Engineering, Chinese University of Hong Kong, Hong Kong, China. (e-mail: xunkuaizhou@cuhk.edu.hk; {gdyang, jhzhang, byzhao}@mae.cuhk.edu.hk; bmchen@cuhk.edu.hk).

layers in the feature fusion stage. Analogously, Chen *et al.* propose TinyDet [29], a lightweight network trained on COCO [30] for small object detection and reducing computational complexity. Similarly, CEAFFOD [31] is proposed by modifying three novel blocks of the single-stage YOLOv4 to detect small objects. However, the aerial small object size poses a much greater complexity than this datasets [26], [30] used to train these methods [24], [25], [27]. Additionally, these methods consume high computational resources. For instance, the footprint of TPH-YOLOv5++ [25] reaches 4.7G and the computational cost of 207.0 GFLOPs. Moreover, TPH-YOLOv5++ achieves a low speed of 11.9 fps on the NVIDIA RTX3090ti GPU. CEAFFOD also suffers from a low inference speed and high memory usage. MFNet [32] is an efficient method using multi-feature fusion. However, the MFNet [32] suffers from lower accuracy in localization and detection. In conclusion, current object detection methods face challenges in achieving a balance between accuracy and speed with low energy consumption.

In real scenarios, multiple challenges such as *complex backgrounds, high energy consumption*, and *slow reasoning speed* may coexist simultaneously, which complicates aerial flying object detection tasks. To address the issues, a small but accurate network (SANet), is proposed for accurate aerial object detection. Specifically, we introduce the spatial attention mechanism to extract the profound interior features, which increases the ability to discriminate in dynamic backgrounds, and the feature extraction framework employs a reduced number of convolutional kernels to reduce the model's parameters and thereby decrease energy consumption. By analyzing the characteristics of ResNet [17] and RepVGG [33] which are commonly utilized to extract features, a feature fusion module named *RepNeck* is proposed to retain and transfer more semantic information to the latter layer by reassembling pixels rather than convoluting features, improving detection effectiveness and efficiency. Finally, we perform systematic ablation studies to validate the efficacy of the SANet. Extensive comparative experiments on public datasets demonstrate that our method outperforms the state-of-the-art approaches. The deployment of our method on the edge-computing device proves its portability and feasibility in real-world applications. These findings collectively support the attainability of a high-accuracy, low-power, low-latency, and low-memory footprint solution for aerial flying object detection in monitoring applications. The qualitative visualization in Fig. 1 demonstrates the effectiveness of our approach in detecting small objects even in complex environments.

II. Aerial Moving Object Detection Pipeline

A. Framework Overview

Our framework in Fig. 2 consists of three stages: feature extraction, feature fusion, and result output. In the feature extraction stage, we enhance training robustness using the mosaic technique for image dataset augmentation. The residual-spatial attention network (i.e. the yellow block) extracts spatial and convolutional features, with the attention layer capturing more profound internal and clearer contour features compared to the initial extraction from layer one. A convolutional downsampling operation follows, obtaining high-level semantic features and augmenting the perception field with the optimized *SPPS*. In the feature fusion stage, our proposed *RepNeck* performs upsampling operations three times (i.e. the pink block) and incorporates addition operations with previous layers of the same channels from the feature extraction stage. Downsampling operations employ reorg layers [34] (i.e. the orange block) instead of convolutional layers, reducing parameters, accelerating inference latency, and preserving more semantic information for subsequent layers. The downsampled feature is concatenated with the upsampled layers, resulting in richer semantic feature information (i.e. feature maps from the fusion stage). Finally, a non-maximum suppression operation generates the output, highlighting crucial feature locations (output feature map). All these operations are seamlessly integrated within an end-to-end network.

The computational consumption measured by BFLOPs in the feature extraction stage is only 5.52 BFLOPs, and the feature fusion stage is 2.56 BFLOPs. The low computational costs of our method indicate its lightweight nature.

B. Attention Module for Spatial Features Extraction

The spatial features are attained by refining and exploring the inter-spatial relationship of features, and the effectiveness has been demonstrated to highlight informative regions [35]. Specifically, after four convolutional operations for the input images, to aggregate channel information from a feature map, the two pooling operations are utilized to generate two 2D maps: $\mathbf{F_{savg}} \in \mathbf{R^{1\times H\times W}}$ and $\mathbf{F_{smax}} \in \mathbf{R^{1\times H\times W}}$. 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. In summary, the computational process of spatial attention is formulated as follows:

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

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

The attention feature map is incorporated into the convolutional layer preceding the attention operation through a residual connection. 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.

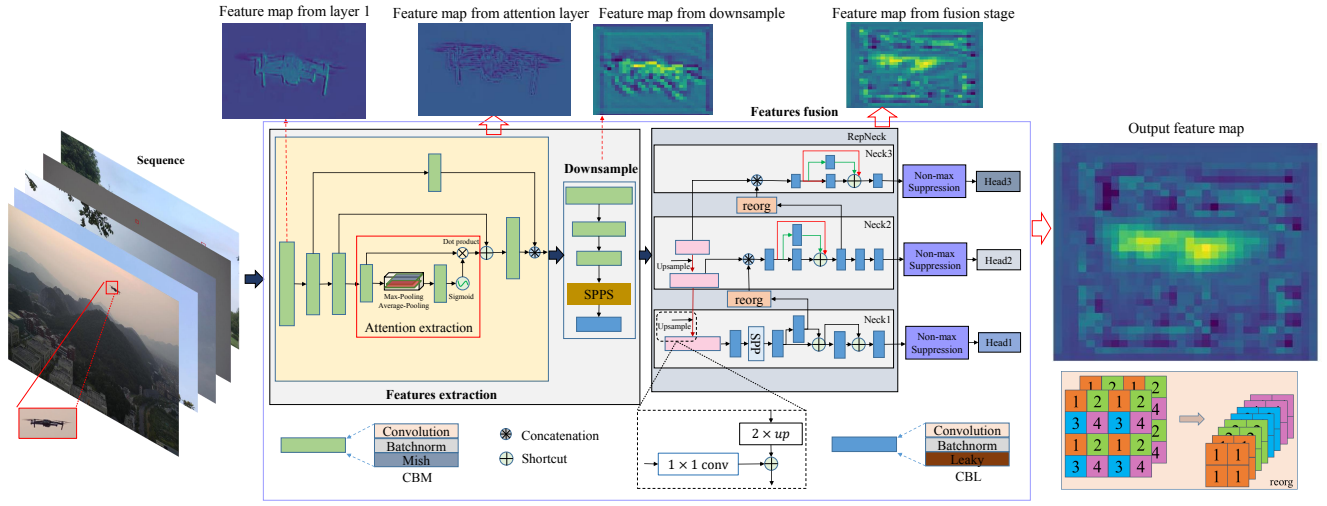


Fig. 2. **SANet Framework:** This framework consists of feature extraction, feature fusion, and result output. The feature extraction component incorporates spatial attention mechanisms to extract more profound features. In the feature fusion component, high-level and low-level semantic features extracted in earlier stages are combined, preserving more semantic information through feature recombination. Finally, the system outputs discriminative features. Please zoom in for the best view.

C. Optimized SPPS for Riching Feature

The complex SPP [36] operation for extracting features influences the inference efficiency. As such, we simplify it with a MaxPool size of 7×7 to replace the original pool operation and integrate the module into the downsample stage. The same operation extension improves efficiency. Concretely, the inputting feature is defined as $\mathbf{F} \in \mathbb{R}^{C\times H\times W}$, the process operation is formulated as:

$$\begin{aligned} &\mathbf{F_c} = Conv(\mathbf{F}), \\ &\mathbf{F_s} = Concate[\mathbf{F_c}, MaxPool_{7\times7}(\mathbf{F_c}), \\ &MaxPool_{7\times7}(\mathbf{F_c}), MaxPool_{7\times7}(\mathbf{F_c})], \\ &\mathbf{F_{spps}} = Conv(\mathbf{F}_s), \end{aligned} \tag{2}$$

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.

D. Proposed RepNeck for Feature Fusion

The statistical data for the aerial object size in TIBNet [37] ranges from 10×5 to 200×80, and most of the object sizes with width and height less than 5% of the image size. Detecting the various scale objects moving against cluttered environments is a fundamental challenge, 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 the lower and the topmost features. However, the bottom-up path method with convoluting operation misses the semantic information. By analyzing the structure characteristic of ResNet [17] and RepVGG [33] shown in Fig. 3(a) and Fig. 3(b), respectively, a common characteristic of both modules has been identified, namely the presence of skip connections, which not only effectively alleviates the issue of gradient vanishing in deep networks but also facilitates the transmission of features extracted from preceding layers to subsequent layers, ensuring the preservation of semantic information. This line of thinking remains applicable to feature fusion as well. As such, we design a feature fusion method shown in Fig 3(c) utilizing the identity 1×1 branches and three branches for enhancing accuracy and efficiency. We orderly denote three necks as *Neck1*, *Neck2*, and *Neck3* from left to right. The reorg modules (i.e. the orange block) are utilized to downsample instead of the convolutional layer among the three necks to preserve the full features. The details are introduced as follows.

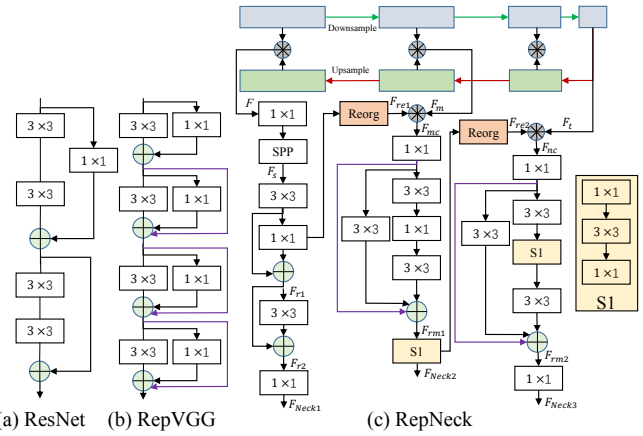


Fig. 3. Sketch of RepNeck architecture. RepNeck has three subpart *Neck1*, *Neck2*, and *Neck3*. We employ a binary and ternary branching structure in the RepNeck.

*1) **Neck1**:* After feature extraction, the upsample operations for the feature map are performed three times from top to bottom (i.e. the purple blocks). Each upsampling operation increases the resolution by a factor of two compared to

the previous level. The five convolutional layers are utilized to extract features between the bottom upsample and the SPP module, and then enter the feature fusion operation. 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, $C1$, $C2$, and $C3$ indicate the channels. 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:

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

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

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

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

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

*3) **Neck3:*** Similar with the *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, and the $\mathbf{F_{re2}} \in \mathbb{R}^{C8\times\frac{H}{4}\times\frac{W}{4}}$ is transformed from the *Neck2*, the $\mathbf{F_{nc}} \in \mathbb{R}^{(C7+C8)\times\frac{H}{4}\times\frac{W}{4}}$ is the concatenation feature map from the $\mathbf{F_t}$ and $\mathbf{F_{re2}}$ with the operation of *reorg*. The operation process is formulated as follows:

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

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

E. Loss Function

We implement three loss functions to optimize our method: (i) Objectness loss. (ii) Classification class. (iii) Localization loss. The feature map $\mathbf{F}$ is treated as an $A\times A$ grid cell, where B bounding boxes are predicted in each cell. The objectness loss and classification loss are calculated based on whether the object is present or not in the cell, as demonstrated in the following equation.

$$\begin{aligned} L_{obj+cls} = &\sum_{k=0}^{A^2}\sum_{j=0}^{B}\mathbb{I}_{ij}^{obj}(cf_i - \hat{cf}_i)^2 + \alpha_{nbj}\sum_{k=0}^{A^2}\sum_{j=0}^{B}(1-\mathbb{I}_{ij}^{obj})(cf_i - \hat{cf}_i)^2\\ &+\sum_{i=0}^{A^2}\sum_{j=0}^{B}\mathbb{I}_{i,j}^{obj}\sum_{c\in classes}\hat{p_i}(c)log(p_i(c)) + (1-\hat{p_i}(c))log(1-p_i(c))), \end{aligned} \tag{6}$$

where, if a bounding box in a grid cell contains the object, the value of $\mathbb{I}_{ij}^{obj}$ is set to 1. If a bounding box in a grid cell does not contain the object, the value of $\mathbb{I}_{ij}^{obj}$ is set to 0. The hyperparameter α_{nbj} is assigned the value of 5. The cf_i and $\hat{cf}_i$ represent the confidence scores of the prediction and ground truth, respectively. The $p_i(c)$ and $\hat{p_i}(c)$ represent the probability score of the predicted class and ground-truth class, respectively.

The localization loss can be defined as:

$$\begin{aligned} &L_{loc} = 1 - IOU + \frac{d^2(b,\hat{b})}{c^2} + \alpha s,\\ &IOU = \frac{|B\cap\hat{B}|}{|B\cup\hat{B}|}, \alpha = \frac{s}{(1-IOU)+s},\\ &s = \frac{4}{\pi^2}(arctan\frac{\hat{\omega}}{\hat{h}} - arctan\frac{w}{h})^2, \end{aligned} \tag{7}$$

where, $\hat{B} = (\hat{x},\hat{y},\hat{\omega},\hat{h})$ and $B = (x,y,\omega,h)$ represent the ground-truth box and predicted box, respectively. The central points of B and $\hat{B}$ are denoted by b and $\hat{b}$, respectively. The function $d(\cdot)$ represents the Euclidean distance, and c represents the diagonal length of the smallest enclosing box that covers both B and $\hat{B}$. The total loss is denoted as follows:

$$Loss_{total} = \lambda_1 L_{obj+cls} + \lambda_3 L_{loc}. \tag{8}$$

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

III. Experiment

A. Implementation Details

*1) **Training phase:*** We train our SANet on the MFNet [32], Det-Fly [38], and TIB-Net [37] using a 1 GPU with the type of GTX 3090, respectively. We choose the stochastic gradient descent optimizer (SGD) with an initial learning rate $\eta_{initial} = 1.3\times10^{-3}$. At the 80% and 90% of setting iterations, the learning rate drops to ten times the previous learning rate. The values of weight decay and momentum are 0.0005 and 0.949, respectively.

*2) **Test phase:*** In our study, Det-Fly [38], we conducted inference time and BFLOPs tests on an RTX 3090 GPU. The testing procedure for the remaining datasets adheres to the guidelines provided in the original paper. In each testing experiment, we input the same original resolution image and measure the accuracy (mAP) on the testing set while evaluating the inference time and BFLOPs.

B. Methods Comparison Results

MFNet. In Table I. Our method demonstrates a significant improvement over the state-of-the-art approaches. Concretely, our method exhibits several advantages, including the lower computational costs measured in BFLOPs, smaller parameters, and higher Intersection over Union (IOU) scores, notably by 2.8% on mAP and 16.7% on IOU over the best one in this evaluation, MFNet-M [32], and 0.7M on model size metric over the MFNet-S. These findings highlight

TABLE I

QUANTIVE BENCHMARKING RESULTS ON MFNET.

Method	Model size (M)↓	FLOPs (B)↓	mAP (%)↑	IOU (%)↑
YOLOv5s [21]	14.9	-	90.6	49.4
MFNet-S [32]	5.2	18.9	90.8	49.1
MFNet-M [32]	9.9	75.3	91.5	51.1
MFNet-L [32]	19.5	157.6	90.9	49.0
SANet (ours)	4.5	8.1	94.3	67.8

[1] The best and second best method is shown in Red and Blue, respectively. The ↑ (↓) indicates that larger (smaller) values lead to a better (worse) performance.

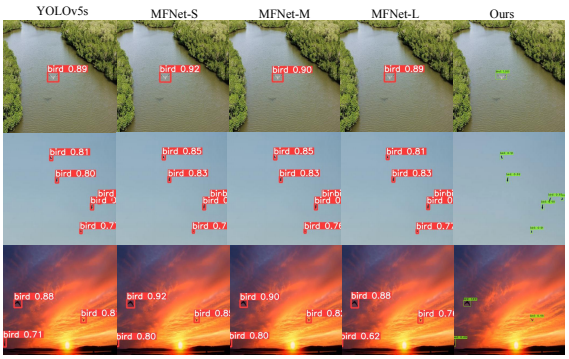


Fig. 4. Qualitative visualization on comparison with state-of-the-art methods. Each row represents the different scenarios. Each column demonstrates the visualization results from YOLOv5s, MFNet-S, MFNet-M, MFNet-L, and SANet from left to right. SANet has higher confidence scores than MFNet, which can demonstrate accurate detection performance. Please zoom in for the best view.

TABLE II

QUANTITIVE BENCHMARKING RESULTS ON TIB-NET

Method	Backbone	mAP (%)↑	Latency (ms)↓	Model size↓
Cascade R-CNN [20]	MobileNet	78.0	164	384.9M
TIBNet [37]	TIB-Net	89.2	290	697.0KB
SANet (ours)	-	90.0	26	4.5M

TABLE III

QUANTITIVE BENCHMARKING RESULTS ON DET-FLY

Method	Image size	mAP (%) ↑	Latency (ms)↓	Model size (M) ↓	Memory footprint (M)↓	FLOPs (B)↓
YOLOv7 [22]	[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 [22]	[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
SANet (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

that the SANet maintains high accuracy and localization quality despite its compact size. Furthermore, the qualitative visualization in Fig. 4 vividly illustrates the superiority of our method compared to the advanced MFNet, particularly in challenging background scenarios. Notably, the samples in Fig. 4 are captured from a distance. Our method consistently achieves superior detection performance.

TIB-Net. The quantitative comparison results are shown in Table II. Our method surpasses Cascade R-CNN [20] in terms of model size, inference speed, and accuracy by a significant margin. Although our approach possesses a bigger model size than the TIBNet [37], we achieve higher accuracy and low latency by a large margin. Due to the extensive usage of attention layers in TIBNet [37] that can slow down the inference speed, which is also reflected in our ablation study in subsection III-C, where we observed the impact of attention layers on inference speed.

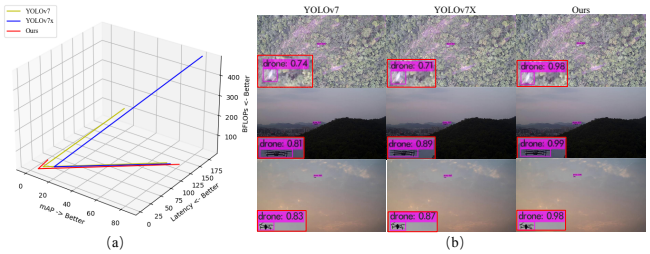


Fig. 5. (a) Trade-off performance of accuracy, latency, BFLOPs on Det-Fly. This left arrow indicates that the closer to the left, the better, and vice versa. (b) Qualitative visualization on comparison with state-of-the-art methods. Each row represents the different scenarios. Each column demonstrates the visualization results from YOLOv7, YOLOv7X, and SANet from left to right. Please zoom in for the best view.

Det-Fly. Table III shows the comparison results on the Det-Fly [38]. Again, the proposed method performs the best performance in terms of both mAP and latency, further demonstrating its robustness. Moreover, the model size of our method is 3% of the advanced method YOLOv7 [22], and the memory footprint and BFLOPs are lower than YOLOv7 [22], demonstrating its low computational complexity. The visualization of the mAP, latency, and BFLOPs of the three methods in Fig. 5(a) demonstrates that our method (i.e. the red curve) has a better 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 usage on edge-computing devices. The qualitative visualization shown in Fig. 5(b) also exhibits the effectiveness of SANet compared to YOLOv7 and YOLOv7X. Our model can accurately locate and detect small flying objects in many challenging situations with a higher detection score (98.5 vs 80.8 on average), such as dark light, and tangled grass.

TABLE IV

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
SANet	**96.3**	**13.6**	**4.5**	**49.0**

C. Ablation Study

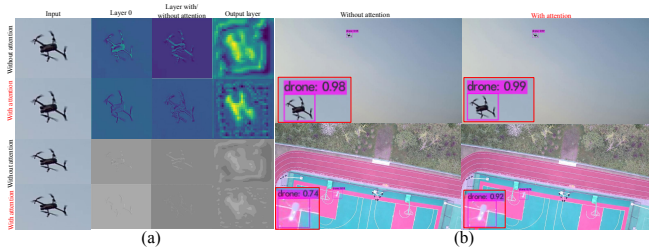
To validate the contributions of aerial object detection and scale estimation to performance improvement, we conduct extensive experiments on Det-Fly [38]. In the following experiments, the input size of the detector in the test phase

is set to 1024 × 1024 pixels. we validate the effectiveness by removing or replacing related modules.

Effect of RepNeck. The experimental results are listed in Table IV. We note that the mAP, model size, and FLOPs decrease. Inversely, the latency increases without *RepNeck*, which means the inclusion of *RepNeck* leads to improvements in accuracy and latency while also introducing a high computational cost. This is because the *RepNeck* can matin feature information with the reorg layers while operating needs more computational cost. The operation is simply a way of recombining information instead of convolution. As such, the latency improves, and model size increases if adding *RepNeck*.

Effect of Attention. From Table IV, The pooling operation does not introduce any extra parameters, except for the one convolutional layer in the attention module, 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 SANet without attention, meaning attention operation enhances the accuracy but degrades the efficiency.

In Fig. 6, We present color and grayscale feature maps with/without integrated attention mechanisms. Rows 1 and 3 show feature maps without an attention mechanism, while rows 2 and 4 depict feature maps with an attention mechanism. The first column corresponds to the input image, and subsequent columns represent outputs from different layers. Our method, with the attention mechanism, extracts more prominent features and apparent contours (column 2) compared to the case without attention. Similarly, sharper internal and contour features are observed (column 3) when the attention mechanism is employed. Moreover, the output feature map from the final layer (last column) demonstrates that our method selectively focuses on relevant object features while ignoring irrelevant information. The qualitative visualization of detection results in Fig. 6(b) shows that the attention mechanism improves the accuracy (95.5 vs 86.0 on average). In summary, the feature map and qualitative visualization analysis confirm the effectiveness of integrating the attention mechanism in our approach.



(a) (b)

Fig. 6. (a) Feature map extracted from different layers. The grayscale feature maps are provided to aid in observing the effects without being influenced by color images. (b) Qualitative visualization. Please zoom in for the best view.

Effect of SPPS. After replacing the SPP with SPPS in the downsampling stage, we note that the speed increases to 13.8ms. 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 SPPS operation does change the number of parameters, the model size, and FLOPs. 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 in YOLOv3-SPP [39] with the SPPS. As shown in Table V, compared with YOLOv3, the SPPS improves the accuracy and latency. Inversely, both accuracy and latency are enhanced compared with YOLOv3-SPP.

TABLE V
ABLATION STUDY OF YOLOV3 ON DET-FLY

Method	mAP (%)↑	Latency (ms)↓
YOLOv3 [39]	87.8	6.3
YOLOv3-SPP [39]	87.5	6.5
YOLOv3-SPPS	**88.4**	**6.4**

D. Deployment for Testing

To demonstrate the deployment capability of SANet on edge-computing devices, we deploy our SANet on an onboard computer, NVIDIA Jetson Orin NX device with 16GB GPU memory and 8 CPU cores. 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**$^{\circ}C$. Compared to the static state, the temperature increase is insignificant when our model runs. Furthermore, the power consumption on the processors is only **2.9W**, confirming that our network satisfies the portability, practicality, and energy-efficient requirements. Table VI shows the temperature and power consumption values of various sensors in two scenarios.

TABLE VI
MODEL DEPLOYMENT TESTING.

Scenarios	**T**$_{CPU}$ (°C)	**T**$_{GPU}$ (°C)	**Power** (W)
Static Scenarios	46.12	43.69	0.582
Running Scenarios (max)	49.03	47.19	3.500
Increment	2.91	3.5	2.918

IV. CONCLUSION

This study proposes a lightweight, energy-efficient approach named SANet for accurately and real-time detecting aerial small flying objects. It demonstrates a satisfactory balance between detection accuracy, inference time, and energy efficiency on public datasets. Our method alleviates resource-constrained surveillance devices' processing burden and showcases edge-computing devices' feasibility and portability. This work collectively supports the idea that achieving a solution for aerial flying object detection in monitoring applications is possible, with high accuracy, low power consumption, low latency, and a small memory footprint. Future work involves extending the method to other flying object domains and improving the detection efficiency.

REFERENCES

[1] K. Feng, W. Li, J. Han, and F. Pan, "Low-latency aerial images object detection for uav," *Unmanned Systems*, vol. 10, no. 1, pp. 57–67, 2022.

[2] S. Ghosh, J. Patrikar, B. Moon, M. M. Hamidi, and S. Scherer, "AirTrack: Onboard deep learning framework for long-range aircraft detection and tracking," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*, 2023, pp. 1277–1283.

[3] H. Cao, J. Zhou, J. Huang, Y. Li, N. C. Meng, R. Cao, Q. Dou, and Y. Liu, "Two-Stage Grasping: A new bin picking framework for small objects," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*, 2023, pp. 2584–2590.

[4] A. Li, S. Ni, Y. Chen, J. Chen, X. Wei, L. Zhou, and M. Guizani, "Cross-modal object detection via uav," *IEEE Transactions on Vehicular Technology*, 2023.

[5] D. Seo and J. Kang, "Collision-avoided tracking control of uav using velocity-adaptive 3d local path planning," *International Journal of Control, Automation and Systems*, vol. 21, no. 1, pp. 231–243, 2023.

[6] C. Zhao, Y. Li, and Y. Lyu, "Event-based real-time moving object detection based on imu ego-motion compensation," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*, 2023, pp. 690–696.

[7] Z. Kaleem and M. H. Rehmani, "Amateur drone monitoring: State-of-the-art architectures, key enabling technologies, and future research directions," *IEEE Wireless Communications*, vol. 25, no. 2, pp. 150–159, 2018.

[8] B. M. Chen, "On the trends of autonomous unmanned systems research," *Engineering*, vol. 12, pp. 20–23, 2021.

[9] K. Tracy, T. A. Howell, and Z. Manchester, "Differentiable collision detection for a set of convex primitives," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*, 2023, pp. 3663–3670.

[10] J. Xie, C. Gao, J. Wu, Z. Shi, and J. Chen, "Small low-contrast target detection: Data-driven spatiotemporal feature fusion and implementation," *IEEE Transactions on Cybernetics*, vol. 52, no. 11, pp. 11 847–11 858, 2021.

[11] 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 Transactions on Cybernetics*, vol. 52, no. 5, pp. 3957–3970, 2020.

[12] H. Wang, J. Zhao, H. Wang, C. Hu, J. Peng, and S. Yue, "Attention and prediction-guided motion detection for low-contrast small moving targets," *IEEE Transactions on Cybernetics*, 2022.

[13] S. Liang and D. Baker, "Real-time background subtraction under varying lighting conditions," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*, 2023, pp. 9317–9323.

[14] Y. Dong, W. Shi, B. Du, X. Hu, and L. Zhang, "Asymmetric weighted logistic metric learning for hyperspectral target detection," *IEEE Transactions on Cybernetics*, vol. 52, no. 10, pp. 11 093–11 106, 2021.

[15] X. Xue, Y. Li, X. Yin, C. Shang, T. Peng, and Q. Shen, "Semantic-aware real-time correlation tracking framework for uav videos," *IEEE Transactions on Cybernetics*, vol. 52, no. 4, pp. 2418–2429, 2020.

[16] J. Becktor, F. Schöller, E. Boukas, and L. Nalpantidis, "Robust uncertainty estimation for classification of maritime objects," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*, 2023, pp. 3154–3160.

[17] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[18] R. Girshick, "Fast R-CNN," in *Proceedings of the IEEE international conference on computer vision (ICCV)*, 2015, pp. 1440–1448.

[19] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 779–788.

[20] Z. Cai and N. Vasconcelos, "Cascade R-CNN: Delving into high quality object detection," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018, pp. 6154–6162.

[21] G. Jocher, "YOLOv5 by Ultralytics," 5 2020. [Online]. Available: https://github.com/ultralytics/yolov5

[22] 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," *arXiv preprint arXiv:2207.02696*, 2022.

[23] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, "Focal loss for dense object detection," in *Proceedings of the IEEE International Conference on Computer Vision (CVPR)*, 2017, pp. 2980–2988.

[24] 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 *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021, pp. 2778–2788.

[25] 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*, vol. 15, no. 6, p. 1687, 2023.

[26] Y. Cao, Z. He, L. Wang, W. Wang, Y. Yuan, D. Zhang, J. Zhang, P. Zhu, L. Van Gool, J. Han *et al.*, "VisDrone-DET2021: The vision meets drone object detection challenge results," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021, pp. 2847–2854.

[27] Y. Jiang, Z. Tan, J. Wang, X. Sun, M. Lin, and H. Li, "GiraffeDet: a heavy-neck paradigm for object detection," *arXiv preprint arXiv:2202.04256*, 2022.

[28] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 4700–4708.

[29] S. Chen, T. Cheng, J. Fang, Q. Zhang, Y. Li, W. Liu, and X. Wang, "TinyDet: accurately detecting small objects within 1 gflops," *Science China Information Sciences*, vol. 66, no. 1, pp. 1–2, 2023.

[30] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft Coco: Common objects in context," in *Proceedings of the European Conference on Computer Vision (ECCV)*. Springer, 2014, pp. 740–755.

[31] A. Elhagry, H. Dai, A. El Saddik, W. Gueaieb, and G. De Masi, "CEAFFOD: Cross-ensemble attention-based feature fusion architecture towards a robust and real-time uav-based object detection in complex scenarios," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*, 2023, pp. 4865–4872.

[32] M. Dil, M. U. Khan, M. Z. Alam, F. A. Orakazi, Z. Kaleem, and C. Yuen, "Safespace mfnet: precise and efficient multifeature drone detection network," *arXiv preprint arXiv:2211.16785*, pp. 1–13, 2022.

[33] X. Ding, X. Zhang, N. Ma, J. Han, G. Ding, and J. Sun, "RepVGG: Making vgg-style convnets great again," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021, pp. 13 733–13 742.

[34] 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 *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 1874–1883.

[35] N. Komodakis and S. Zagoruyko, "Paying more attention to attention: improving the performance of convolutional neural networks via attention transfer," in *International Conference On Learning Representations (ICLR)*, 2017.

[36] K. He, X. Zhang, S. Ren, and J. Sun, "Spatial pyramid pooling in deep convolutional networks for visual recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 37, no. 9, pp. 1904–1916, 2015.

[37] H. Sun, J. Yang, J. Shen, D. Liang, N. Liu, and H. Zhou, "TIBNet: Drone detection network with tiny iterative backbone," *IEEE Access*, vol. 8, pp. 130 697–130 707, 2020.

[38] 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 and Automation Letters*, vol. 6, no. 2, pp. 1020–1027, 2021.

[39] J. Redmon and A. Farhadi, "YOLOv3: An incremental improvement," *arXiv preprint arXiv:1804.02767*, 2018.