Research on UAV Target Detection Based on APFU-YOLOv10

Hongpei Zhang
School of Computer Science and Engineering
Xi'an Technological University
Xi'an, 710021, Shanxi, China
E-mail: 2464094384@qq.com

Yijian Zhang
School of Computer Science and Engineering
Xi'an Technological University
Xi'an, 710021, Shanxi, China
E-mail: 2278662342@qq.com

Bailin Liu
School of Computer Science and Engineering
Xi'an Technological University
Xi'an, 710021, Shanxi, China
E-mail: 498194312@qq.com

Zhixuan Zhao
School of Computer Science and Engineering
Xi'an Technological University
Xi'an, 710021, Shanxi, China
E-mail: 18535049831@163.com

Wenfei Sheng
School of Mechantronic Engineering
Xi'an Technological University
Xi'an, 710021, Shanxi, China
E-mail: 729784334@qq.com

Feng Xiong
School of Computer Science and Engineering
Xi'an Technological University
Xi'an, 710021, Shanxi, China
E-mail: 1023465977@qq.com

***Abstract*—In UAV-captured images, the high density of objects and the large proportion of small targets pose significant challenges to YOLO-based object detection algorithms. This study presents an enhanced object detection framework derived from the YOLOv10s architecture, aiming to achieve superior detection accuracy. First, an Adaptive Progressive Feature Unification (APFU) module is proposed to effectively integrate multi-level feature representations, ensuring a balanced fusion of high-level semantic information from low-resolution features and fine-grained spatial details from high-resolution features. Second, a Feature Enhancement and Attention (FEA) module is introduced to adaptively recalibrate feature responses, emphasizing informative features while suppressing irrelevant noise and interference. Finally, based on these modules, the APFU-YOLOv10 network is built to effectively improve the network's perception ability of objects at different scales. Experimental results on the VisDrone dataset demonstrate the superior performance of the proposed algorithm: mAP@0.5 increased from 42.6% to 43.5%, a relative improvement of approximately 2.11%; mAP@0.5:0.95 improved from 25.4% to 26.2%, a relative increase of about 3.15%; recall improved from 0.410 to 0.416, further reducing missed detections and enhancing object coverage. The method achieves significant improvements in detection accuracy under medium to high IoU thresholds, validating the effectiveness of multi-scale feature fusion and adaptive attention mechanisms in small object detection for UAV imagery.**

Keywords- Yolov10s; Multi-Scale Fusion; Adaptive Mechanisms

I. INTRODUCTION

Recent technological advancements in unmanned aerial vehicle (UAV) systems, particularly in onboard hardware and flight control algorithms, have enabled their broad adoption across multiple disciplines. These sophisticated platforms are now routinely utilized for geological mapping, precision farming, weather monitoring, and tactical reconnaissance applications. This has resulted in an exponential growth of UAV-acquired image data, making object detection in UAV imagery an important research focus in the field of computer vision [1]. Compared with traditional ground-level scenes, UAV imagery presents several challenges, including high target

density, a large proportion of small objects, and significant scale variation. UAV-based aerial detection missions often demand rapid and accurate target identification in complex environments, thus requiring detection algorithms with high real-time performance and precision. Due to altitude constraints, small objects occupy a large portion of UAV images, resulting in weak visual features and increased detection difficulty [10]. Additionally, varying UAV flight attitudes introduce large differences in object scale [9], which negatively affect model robustness. The frequent occurrence of occlusion further reduces the stability of continuous detection.

Traditional target detection methods usually rely on manually created features and employ sliding windows or region proposal techniques. They are not suitable for complex drone scenarios and have limitations in terms of accuracy and efficiency.The advancement of deep learning has propelled object detection into a phase of rapid evolution [2]. Mainstream detection algorithms are generally categorized into two types: two-stage detectors, such as the R-CNN series [3], which extract and classify candidate regions for high-precision detection but suffer from slow inference speed and are unsuitable for real-time applications; and one-stage detectors, such as the YOLO series [4], which directly regress object positions and classes with a streamlined structure, achieving high inference speed and making them well-suited for deployment on UAVs and other edge devices. The YOLO series adopts an end-to-end detection framework that strikes a balance between detection speed and accuracy, making it a prominent solution for small object detection tasks.

Despite the continued advancements of the YOLO series and its ability to balance detection precision and inference efficiency, several limitations persist in multi-scale feature representation, adaptive feature extraction, and performance in scenarios involving densely packed small objects. YOLOv5 [5] introduced a multi-scale feature fusion mechanism to improve the recognition of targets at different sizes; however, semantic consistency and contextual integration across feature levels remain insufficient. YOLOv7 [6] made improvements in network architecture and training strategies to enhance model robustness, but it still suffers from limited feature expressiveness when faced with complex images containing dense or scale-variant objects. YOLOv8 [7] incorporated channel and spatial attention mechanisms to strengthen the model's ability to focus on critical regions. Although this improved adaptability to some extent, its detection stability in high-density or cluttered background conditions is still suboptimal. YOLOv10 [8] preserved the end-to-end efficiency of its predecessors while further reducing model size to meet lightweight deployment requirements. Versions such as YOLOv10s demonstrated significant speed advantages; however, their multi-scale feature fusion strategies remain relatively simple, limiting the ability to capture fine-grained information across scales. As a result, detection accuracy in typical UAV aerial imagery scenarios, which often contain densely distributed small objects, declines—exposing the model's inadequate adaptation to complex scale structures and weak perception of small targets. Therefore, improving the model's capacity for multi-scale information representation and adaptive processing remains a key direction for further optimizing YOLO-based detection frameworks.

To overcome these challenges, we present APFU-YOLOv10, an improved object detection framework adapted from YOLOv10s, optimized specifically for UAV-captured aerial imagery. The key innovations of this study include:

(1) To alleviate the limitations of current detection models in multi-scale information fusion, we design a feature enhancement module named APFB. This module introduces a three-stage progressive combination of max-pooling and average-pooling operations, along with learnable adaptive fusion coefficients, enabling dynamic aggregation of features under various receptive fields. In parallel, a multi-scale grouped convolution structure [12] is applied to the pooled results, further strengthening the expression of fine-grained features. This module significantly enhances the model's robustness in detecting objects with large scale variation and exhibits

superior detail preservation and regional response capabilities under conditions of densely packed small objects and complex background interference. It effectively compensates for the limited low-level semantic modeling capabilities of the backbone network.

(2) To address the lack of spatial awareness in densely packed small-object regions, we design the FEA module, which integrates a spatial attention mechanism with a lightweight feedforward enhancement structure to improve the modeling of object positional information. This module adopts a channel-splitting strategy, dividing the input features into static-preserving and dynamic-enhancing branches. The latter utilizes a multi-head attention mechanism to capture spatial contextual dependencies, followed by a feedforward network to further reinforce regional feature responses. This design enhances the model's sensitivity to local structures and its ability to identify key regions, achieving improved detection accuracy and robustness, particularly in scenarios involving dense and small-scale targets. It provides structural support for spatial distribution awareness in complex environments.

(3) To address the performance bottlenecks of existing YOLOv10 models in multi-scale feature fusion, adaptive enhancement, and small-object detection, this paper proposes a modified detection architecture named APFU-YOLOv10, built upon the YOLOv10 backbone. APFU-YOLOv10 retains the end-to-end efficiency of YOLOv10 while significantly improving its capability in fine-grained feature modeling and spatial structure representation. It demonstrates superior robustness and detection accuracy in typical UAV aerial imaging scenarios.

II. YOLOv10-Based Object Detection Algorithm

YOLOv10 represents the most recent advancement in the YOLO series architecture, incorporating innovative modules while maintaining the proven advantages of previous iterations. This versatile framework demonstrates capabilities beyond conventional object detection, including image classification, semantic segmentation, and object tracking applications. Its exceptional computational efficiency and adaptable architecture have established YOLOv10 as a significant focus of research within the computer vision community.

In this study, the YOLOv10s model is selected as the baseline due to its balanced performance. The YOLOv10s framework employs a tripartite structural composition, comprising: (1) a feature extraction backbone network, (2) an intermediate feature fusion neck, and (3) a detection head for final prediction outputs.

The feature extraction backbone utilizes a Darknet-53 inspired architecture enhanced with YOLOv8's CSPLayer_2Conv module to facilitate residual connections. A key innovation is the implementation of spatially and channel-wise decoupled downsampling (SCDown), which substantially enhances computational efficiency during dimensionality reduction.

To mitigate architectural redundancy resulting from uniform basic blocks in conventional YOLO architectures, YOLOv10 implements an efficiency-accuracy co-optimization framework. This innovative approach integrates: (1) a Compact Inverted Bottleneck (CIB) module for streamlined feature transformation, and (2) an adaptive hierarchical block allocation scheme, collectively optimizing computational performance while maintaining detection accuracy.

Furthermore, the model integrates the Spatial Pyramid Pooling-Fast (SPPF) module from YOLOv8 to normalize feature vector sizes across different scales. Following the SPPF, YOLOv10 adds a Fast Efficient Attention (FEA) module, which enhances performance while alleviating the typical computational burden associated with attention mechanisms.

The Neck component focuses on feature fusion. The architecture incorporates a bidirectional Path Aggregation Network (PANet) to facilitate cross-scale feature integration through hierarchical top-down and bottom-up propagation pathways.

The detection head architecture employs depthwise separable convolutional layers to construct efficient classification modules, achieving substantial parameter reduction while

maintaining detection performance.It also incorporates a dual-label assignment strategy, combining one-to-many and one-to-one label assignments. During training, both assignments contribute to learning; however, during inference, only the one-to-one assignment is used, allowing for end-to-end deployment without the need for Non-Maximum Suppression (NMS) and without additional inference cost.

The final prediction module consists of three detection heads at different scales to accommodate objects of various sizes.

Compared to other variants in the YOLOv10 series, YOLOv10s achieves a better balance between speed and accuracy. With only 7.2 million parameters, it delivers significantly improved accuracy over the lightweight YOLOv10n, while requiring far fewer computational resources than YOLOv10m, which, although more accurate, is less efficient. Consequently, YOLOv10s achieves an optimized balance between computational efficiency and detection accuracy. The overall framework of YOLOv10 is depicted in Figure 1.

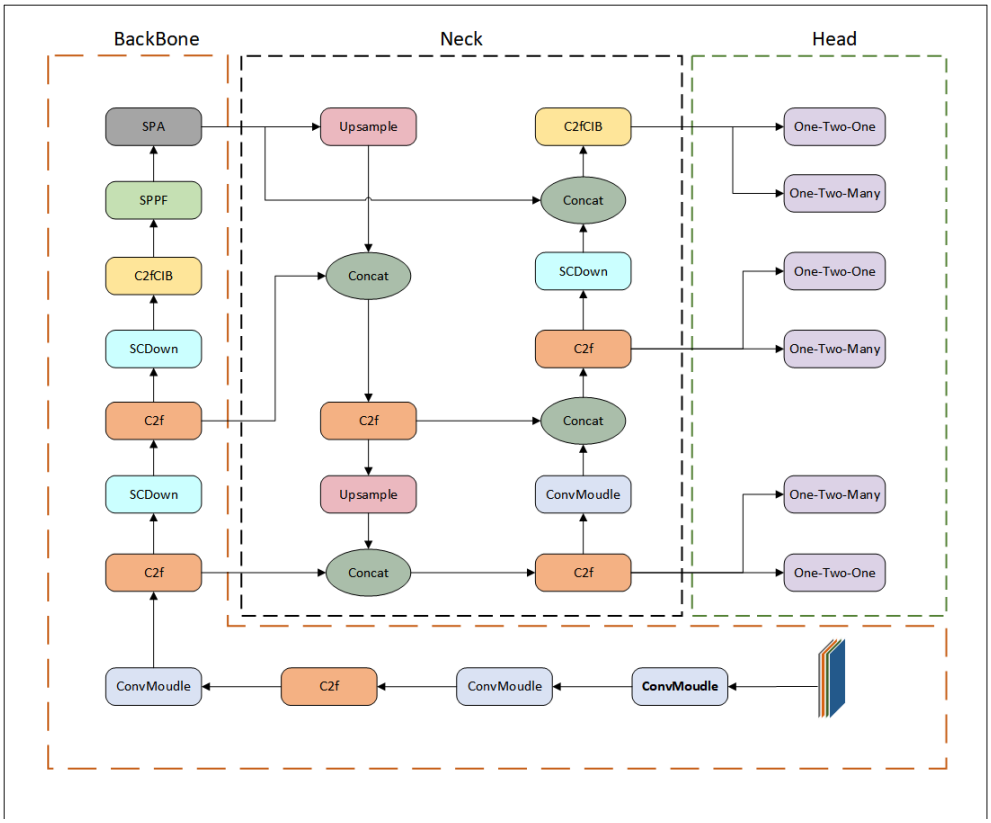

Figure 1. YOLOv10 model structure

III. APFU-YOLOv10 Algorithm

This study introduces APFU-YOLOv10, an advanced object detection framework derived from YOLOv10, which is specifically optimized for handling multi-scale detection challenges. The developed architecture enhances spatial localization through specialized coordinate-aware modules and incorporates adaptive feature refinement mechanisms with learnable weighting coefficients.

The proposed architecture initially develops an Adaptive Feature Pyramid Boosting (APFB) module to perform feature map recalibration through dual mechanisms: (1) selective amplification of semantically rich regions, and (2) dynamic suppression of background interference. This feature optimization process enables more

robust characteristic extraction, subsequently delivering enhanced discriminative representations for downstream network components.

The Feature Enhancement Attention (FEA) module is subsequently employed to facilitate cross-scale feature integration, adaptively fusing high-level semantic representations from coarse feature maps with detailed spatial information from fine-resolution features. This hierarchical fusion mechanism enhances the discriminative power of multi-scale feature representations through learned attention weights. The FEA module adaptively adjusts the importance of these features by learning spatial and contextual dependencies, emphasizing salient object characteristics and improving the network's ability to detect objects across varying scales.

Building upon this framework, we introduce the APFU-Neck to strengthen multi-scale feature fusion, enhancing the model's capability to detect objects across varying scales. The Head component maintains the original YOLOv10 detection head structure. The complete architecture of the proposed network is illustrated in Figure 2.

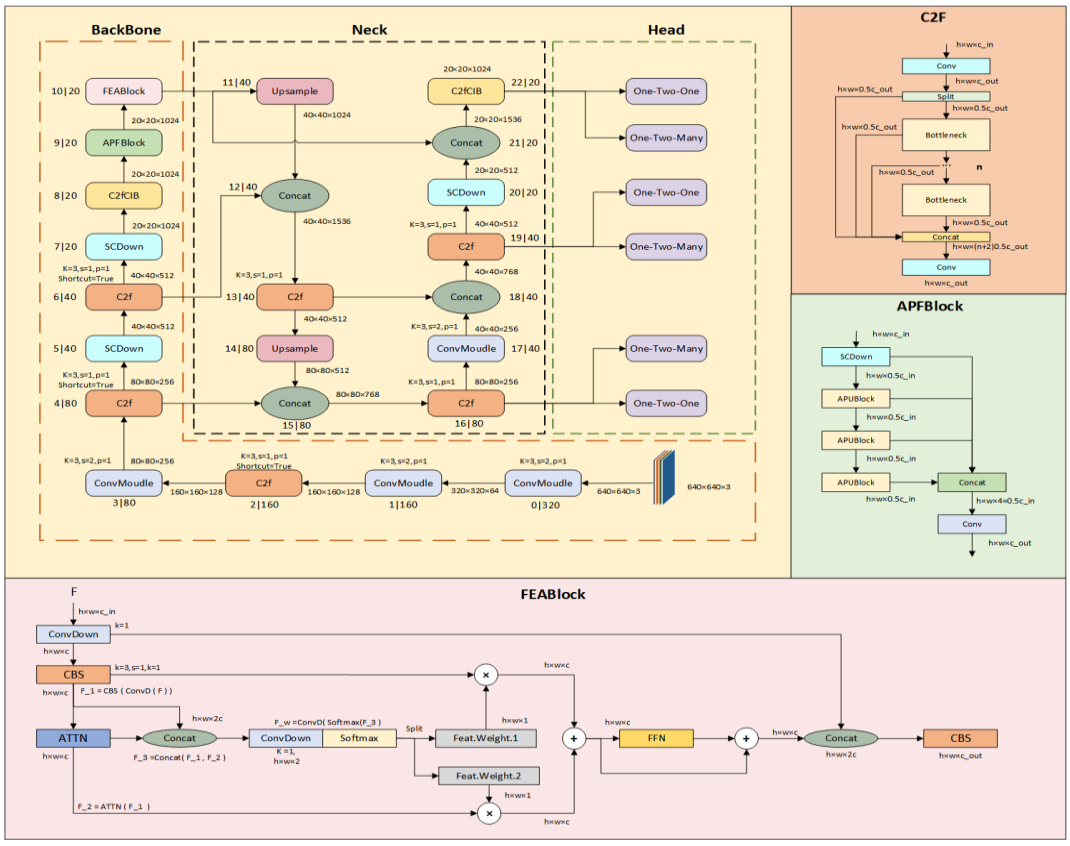

Figure 2. APFU-YOLOv10 model structure

A. Adaptive Pooling Fusion Block

In the process of feature extraction in deep learning models, effectively integrating multi-scale information to enhance representational capacity has long been a key area of research. Traditional Spatial Pyramid Pooling-Fast (SPPF) enhances the model's receptive field by applying fixed-scale max pooling operations. However, this approach has several limitations in feature fusion:

Although max pooling emphasizes salient features, it can simultaneously discard background information, leading to incomplete feature representations.

The feature concatenation is fixed and does not dynamically adjust the contributions of different scales according to the input features.

The module only performs simple concatenation after pooling, without deep fusion, which limits the complementary interactions among features.

To address these issues, this paper proposes a novel Fusion-Driven Adaptive Pooling Module (APFB). This module introduces an adaptive fusion pooling mechanism that enables more accurate feature extraction and fusion while maintaining computational efficiency. The core component, AFPB, is illustrated in Figure 3. It integrates convolutional layers, max pooling, average pooling, and learnable fusion weights. Through adaptive learning of parameters (α, β, and γ), the module achieves weighted information fusion, enabling the model to dynamically emphasize important features across different scales.

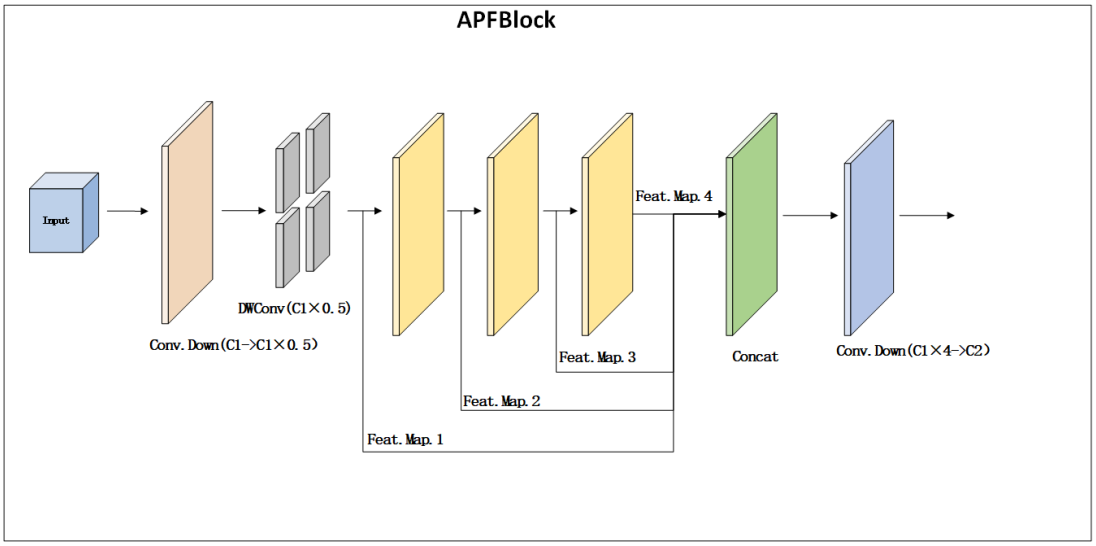

Figure 3. APFBmodel structure

Both APFB and SPPF are modules designed for multi-scale feature processing, but they differ significantly in their design philosophy and functional implementation. SPPF (Spatial Pyramid Pooling-Fast) enhances multi-scale representation by applying spatial pyramid pooling with kernels of different sizes to the input feature maps, followed by concatenation of the pooled outputs. This method is simple and computationally efficient, making it suitable for tasks requiring information extraction across multiple scales.

In contrast, the Adaptive Pooling Fusion Block (APFB) builds upon this foundation by introducing an adaptive fusion mechanism. Through learnable parameters, it dynamically adjusts the fusion weights of different pooling operations, thereby optimizing the feature fusion process. The proposed architecture employs a hierarchical pooling fusion mechanism as its fundamental component. Initially, the input feature map undergoes channel dimension reduction through a 1×1 convolutional layer, decreasing channel capacity by 50%. Subsequently, a 3×3 convolutional layer processes the compressed features to capture higher-level representations. These features are then fed into a three-stage pooling fusion process. Each stage combines max pooling and average pooling, with learnable parameters controlling the weighted fusion. This allows the network to dynamically balance its attention between local and global information.

After the three-level fusion, the outputs from each stage are concatenated and passed through another 1×1 convolution for channel integration, resulting in an optimized feature representation.

As shown in Figure 4, the Adaptive Pooling Unit (APU) serves as the core component of the APFB module. Each pooling fusion stage in APU follows an identical structure: the input features undergo both max pooling and average pooling to capture strong local activations and smoothed contextual features, respectively. Then, a learnable parameter controls the fusion ratio between the two, enabling adaptive expression of different scale features. The fused features are processed in parallel by two convolutional branches—one employing a 7×7 convolution for a large receptive field, and the other using a 3×3 convolution for fine-grained details. The outputs of both convolutions are element-wise added and passed through the SiLU activation function for non-linear transformation, thereby enhancing the expressiveness of the resulting feature map.

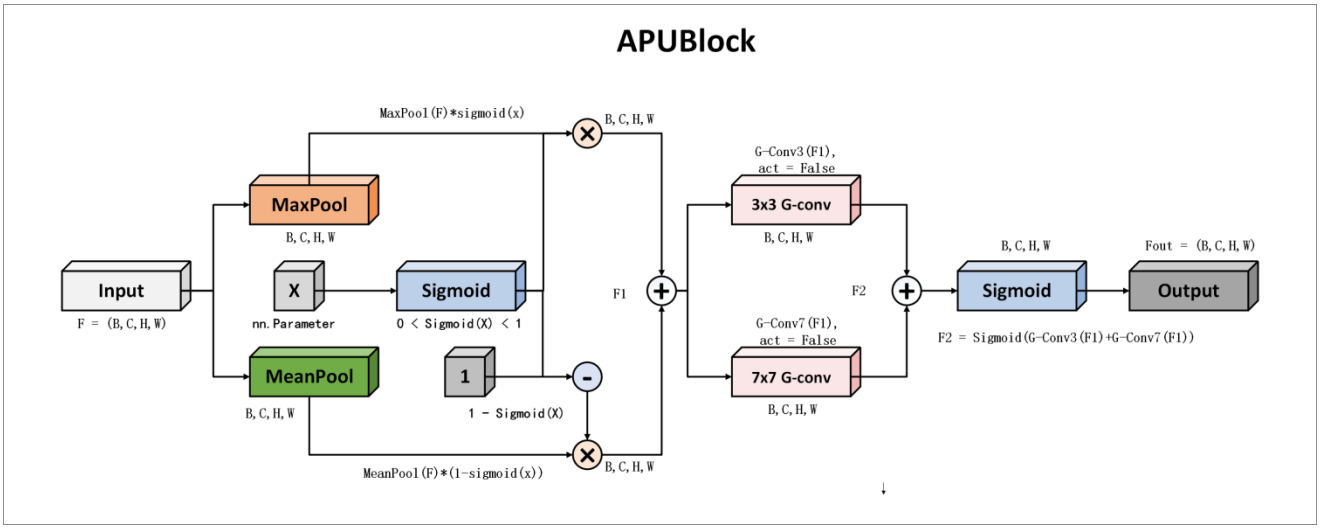

Figure 4. APU model structure

After completing the multi-level pooling fusion, the APFBlock module concatenates features from different scales and then applies a 1×1 convolutional layer for channel compression, ultimately producing the final output feature map. This design enhances feature representation capability while ensuring computational efficiency. The following provides a detailed description of this process along with its mathematical formulation.

The APFBlock module processes an input feature map x, where H and W represent spatial dimensions and C_1 denotes channel depth. The initial processing stage applies a 1×1 convolutional layer for channel reduction, generating an intermediate feature map F_1 with halved channel dimensionality.

$$F_1 = Conv_{1\times1}(F_{in}; C_1; \frac{C_1}{2}) \quad (1)$$

Next, the feature map F1 is processed by a 3×3 convolutional layer to further extract deep features, resulting in the feature map F2.

$$F_2 = Conv_{3\times3}(F_1; C_1; \frac{C_1}{2}) \quad (2)$$

The innovation of the APFB module in pooling lies in the weighted fusion of max pooling and average pooling. The weighting ratios for the pooling methods are controlled by learnable parameters α, β, and γ, which are mapped to the range [0, 1] through a Sigmoid function, enabling the model to adaptively adjust the pooling strategies.

First-level pooling fusion, During the first-level pooling fusion, the feature map F2 is processed simultaneously by max pooling and average pooling operations. The weighted ratio between max pooling and average pooling is governed by the parameter α. The fused pooled feature map is given by FPool1:

$$F_{\text{pool1}} = \sigma(\alpha)\ \text{MaxPool}\ (\text{F}_2) + (1-\sigma(\alpha))\,\text{AvgPool}(\text{F}_2) \tag{3}$$

Where σ(x) denotes the Sigmoid activation function, which maps the parameter α\alphaα to the range [0,1]. Subsequently, the pooled feature map undergoes two convolution operations for further feature extraction:

$$F_{conv1} = \text{C}onv_{7\times7}(\text{F}_{pool1}) + \text{Conv}_{3\times3}(\text{F}_{pool1}) \tag{4}$$

The feature map Fconv1 obtained after the convolution operations is passed through the SiLU activation function for non-linear transformation:

$$F_{conv1} = SiLu(\text{F}_{conv1}) \tag{5}$$

In the second level of pooling fusion, the weighting ratio for the fused pooling operations is controlled by the parameter β. The pooling and convolution procedures are similar to those in the first layer, and the computations are as follows:

$$F_{\text{pool2}} = \sigma(\beta)\ \text{MaxPool}\ (\text{F}_{conv1}) + (1-\sigma(\beta))\,\text{AvgPool}(\text{F}_{conv1}) \tag{6}$$

The pooled feature map similarly undergoes two convolution operations to extract deeper features and is then passed through the SiLU activation function for non-linear transformation.

$$F_{conv2} = \text{C}onv_{7\times7}(\text{F}_{pool2}) + \text{Conv}_{3\times3}(\text{F}_{pool2}) \tag{7}$$

$$F_{conv2} = SiLu(\text{F}_{conv2}) \tag{8}$$

The third-level pooling fusion adopts the same pooling strategy, with the fusion ratio controlled by the parameter γ. The corresponding formula is as follows:

$$F_{\text{pool3}} = \sigma(\gamma)\ \text{MaxPool}\ (\text{F}_{conv3}) + (1-\sigma(\gamma))\,\text{AvgPool}(\text{F}_{conv3}) \tag{9}$$

The pooled feature map undergoes two convolutional operations for further feature extraction, resulting in the feature map Fconv3, which is then activated using the SiLU activation function:

$$F_{conv2} = \text{C}onv_{7\times7}(\text{F}_{pool3}) + \text{Conv}_{3\times3}(\text{F}_{pool3}) \tag{10}$$

$$F_{conv3} = SiLu(\text{F}_{conv3}) \tag{11}$$

In the three stages of pooling fusion, the feature maps from each stage are concatenated together to form a multi-scale feature representation:

$$F_{concat} = \text{C}oncat(\text{F}_2; F_{conv1}; F_{conv2}; F_{conv3}) \tag{12}$$

Finally, the concatenated feature map is passed through a 1×1 convolutional layer for channel compression, resulting in the final output feature map:

$$F_{out} = Conv_{1\times1}(\text{F}_{concat}; 4\times\frac{C_1}{2}; C_2) \tag{13}$$

The APFB module achieves adaptive pooling fusion by introducing a weighted combination of max pooling and average pooling, where the pooling ratios are controlled by learnable parameters α, β, and γ. This design allows APFB to dynamically adjust the pooling strategy based on the input features, thereby enabling more effective multi-scale information integration. In addition, the fused feature maps undergo further convolutional and activation processing to enhance the model's feature representation capability. Finally, the output feature maps from multiple pooling fusion layers are concatenated and passed through a channel compression layer to generate the final multi-scale feature representation, which significantly improves the model's discriminative power and robustness.

B. Feature Extraction Attention Block

Remote sensing images often contain a large amount of fine-grained information, including targets of various categories such as roads, buildings, and vegetation. For these fine-grained targets, traditional convolutional networks may struggle to effectively capture subtle features, leading to reduced detection accuracy. To address this issue, this paper introduces a self-developed FEA (Feature Enhancement Aggregation) module. In the design of this module, multi-stage feature weighting is employed, enabling the model to enhance features at different stages. This allows for more accurate recognition of small or complex targets. The structure of the module is illustrated in Figure 5.

The FEA module is a feature extraction method that combines convolutional operations with a self-attention mechanism, aiming to enhance the model's ability to focus on spatial features and thereby improve feature learning effectiveness. First, the input feature map is passed through a 1×1 convolutional layer to reduce the number of channels from C1 to a smaller dimension C, which effectively decreases computational cost while preserving critical features. Then, a 3×3 convolution is applied to the feature map for deeper processing, capturing more detailed spatial information. Subsequently, a self-attention mechanism is employed to weight the convolutional features, generating an enhanced weighted feature map. This process enables the model to dynamically adjust its focus on different regions based on the importance of the features.

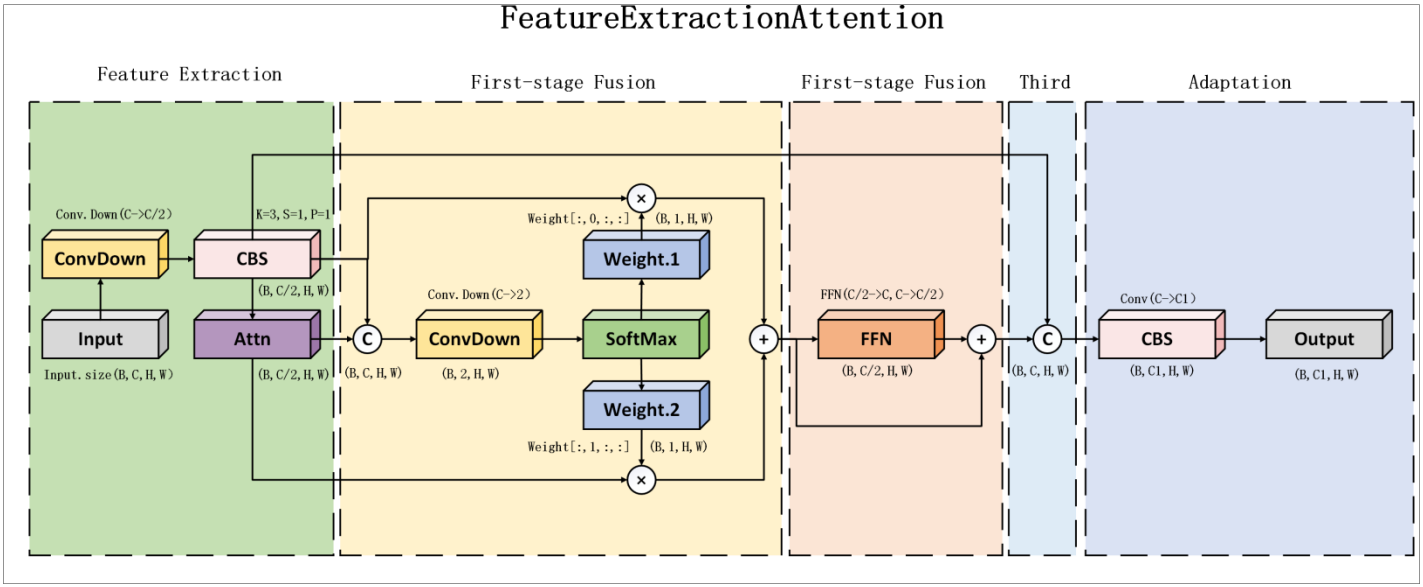

Figure 5. Structure of FEA module.

After concatenating the weighted feature map with the original feature map, a 1×1 convolution is applied to generate weighting coefficients, followed by a softmax operation to enable effective weighting between the original features and the attention-enhanced features. In this way, the FEA module dynamically fuses the original and attention-weighted features based on the importance of each feature. The resulting weighted feature map is then passed into a feedforward network for further processing. After two convolutional layers, the feature map is restored to its original number of channels. Finally, a 1×1 convolutional layer is used to project the processed feature map into the final output. This feature extraction strategy not only enhances the model's ability to capture key information but also improves performance in

complex tasks—especially those that require precise spatial understanding—making it highly effective in computer vision applications. The following section provides a detailed description of this process along with the corresponding mathematical formulations.

The input feature map, denoted as x, where B indicates the batch size, C_1 represents the input channel dimension, and H × W corresponds to the spatial dimensions, initially undergoes channel transformation through a 1×1 convolutional operation. This processing yields an intermediate feature representation a, where C_2 denotes the transformed channel dimension while preserving the original spatial resolution.

$$a = Conv_1(x) \tag{14}$$

Here, $Conv_1$ is a 1×1 convolution operation that maps the input channel dimension C_1 to a smaller channel dimension C, which helps reduce computational complexity. Subsequently, the module applies a 3×3 convolutional layer to further process the feature map, generating deeper spatial features $a' \in R^{B\times C\times H\times W}$.

$$a' = Conv_3(a) \tag{15}$$

Next, the FEA module applies an attention mechanism to assign weights to the convolutional feature map. Let the output of the attention mechanism be denoted as b, that is:

$$b = Attn(a') \tag{16}$$

The self-attention mechanism computes the correlations between different positions within the feature map to assign weights accordingly, thereby highlighting regions containing important information. By incorporating self-attention, the model can dynamically adjust the weights of the feature map based on the contextual information of each position.

Next, the original feature map a' and the attention-enhanced feature map b are concatenated along the channel dimension, resulting in the fused feature map:

$$out1 = Cat(a', b) \tag{17}$$

The concatenated feature map is processed through a 1×1 convolutional layer to generate a weight coefficient map out1w

$$out1w = Conv_5(out1) \tag{18}$$

a softmax operation is applied to out1w to compute the weight coefficients for each channel.

$$out1w = Soft\max(out1w, dim=1) \tag{19}$$

Through the softmax operation, two channel-wise weighting coefficients are obtained, corresponding to the weighted values of feature maps A and B, respectively.

The attention weights obtained through the Softmax operation are used to perform weighted summation on the feature maps a' and b. Specifically, the weighted features are denoted as b_1 and b_2, respectively. Finally, the fused feature representation is obtained by combining these two weighted features.

$$b_1 = a' \times out1w_{[:,0,:,:]} \tag{20}$$

$$b_2 = a' \times out1w_{[:,1,:,:]} \tag{21}$$

$$b = b_1 + b_2 \tag{22}$$

This weighting process facilitates the dynamic adjustment of the contributions from different regions based on feature importance, thereby enhancing the representational capacity of the feature map.

The weighted feature b is subsequently fed into a Feed-Forward Network (FFN), which consists of two 1×1 convolutional layers. The first layer expands the number of channels to 2C, while the second layer compresses it back to C. This process is designed to further refine and extract informative features.

$$b_3 = FFN(b) \tag{23}$$

Through this process, the Feed-Forward Network enhances the representational capacity of the feature map.

The feature b, processed by the Feed-Forward Network, is added to the weighted fused feature to obtain the final output feature map.

$$out = b + b_3 \tag{24}$$

Finally, the fused feature is combined with the shallow features and passed through a 1×1 convolutional layer to map the output feature map to the desired number of output channels C2.

$$output = Conv_1(Cat(a', out)) \tag{25}$$

The Feature Enhancement and Attention (FEA) module synergistically combines convolutional processing with self-attention mechanisms, significantly improving feature representation learning. This hybrid architecture enhances spatial feature discrimination, particularly in scenarios requiring fine-grained structural analysis.

IV. Experimental Design and Result Analysis

A. Experimental Environment and Dataset

This study was conducted on a workstation running Windows 11 Professional, featuring an Intel® Core™ i7-10700 processor (base clock: 2.90 GHz) accelerated by an NVIDIA Quadro M6000 graphics card with 12GB GDDR5 memory. The deep learning environment is based on PyTorch version 2.2.2, and the programming language used is Python 3.12.

The proposed method is evaluated on the VisDrone benchmark dataset [11], a large-scale aerial imagery collection developed by Tianjin University's AISKYEYE team. This drone-captured dataset has become a standard evaluation platform for various vision tasks including detection, tracking, and classification in UAV-based applications. The VisDrone dataset contains 10,209 static images, of which 6,471 images are allocated for training, 3,190 for testing, and 548 for validation. The dataset features complex backgrounds, a wide range of object sizes including small-scale objects, and encompasses 10 object categories: car, pedestrian, bus, bicycle, tricycle, awning-tricycle, truck, van, person, and motorbike. Each image is accompanied by detailed annotations including object categories, bounding box locations, and visibility information. The images are collected under varying temporal and environmental conditions, accounting for variations in lighting, weather, and other factors that affect object detection and tracking. The VisDrone dataset is widely used in the fields of deep learning and computer vision and serves as a standard benchmark for evaluating object detection and tracking algorithms.

B. Evaluation Metrics

This study employs standard object detection metrics including precision-recall measures (P, R, AP), mAP (the predominant detection benchmark), model size, and GFLOPs. The mAP metric is particularly emphasized due to its established role in detection performance evaluation. It is calculated based on both Precision and Recall, with the computation steps shown in Equations (26) to (29).

$$P = \frac{TP}{TP + FP} \tag{26}$$

$$\mathrm{R} = \frac{TP}{TP + FN} \tag{27}$$

$$AP = \int_0^1 P(R)dR \tag{28}$$

$$mAP = \frac{1}{N}\sum_{i=1}^{n} APi \tag{29}$$

The evaluation metrics are defined as: True Positives (TP) count correct detections, False Positives (FP) tally misclassified backgrounds, and False Negatives (FN) record missed targets, with N being the category count - fundamental for precision-recall computation in object detection. The average precision for the iii-th category, denoted as APi, is calculated based on Precision (P) and Recall (R).

The number of parameters measures the model size, while computational complexity quantifies the model's computational cost. Smaller parameter counts and lower computational demands indicate reduced resource requirements, facilitating lightweight deployment.

C. Algorithm Detection Results

The detection results of YOLOv10 and the improved APFU-YOLOv10 on the test set are presented in Table 1.

As shown in Table, compared to the original YOLOv10 model, the APFU-YOLOv10 achieves a significant improvement in detection performance while maintaining a lightweight network architecture.

TABLE I. COMPARISON BEFORE AND AFTER ALGORITHM IMPROVEMENT

Class	***YOLOv10***	***APFU-YOLOv10***
Layers	237	253
Parameters	7.22M	7.70M
GFLOPs	21.4	21.8
Box Precision (P)	0.530	0.539
Recall (R)	0.405	0.417
mAP@0.5	0.426	0.435
mAP@0.5:0.95	0.258	0.262
Inference Speed	7.8ms	8.2ms

The improved model has 7.70 million parameters and a computational cost of 21.8 GFLOPs, with an inference speed of 8.2 ms per image, satisfying real-time detection requirements. In terms of accuracy metrics, APFU-YOLOv10 attains an mAP@0.5 of 0.435 and an mAP@0.5:0.95 of 0.262, with recall increased to 0.417. These results indicate that the model outperforms the original in both precision and completeness of object recognition, demonstrating more stable and reliable overall detection performance.

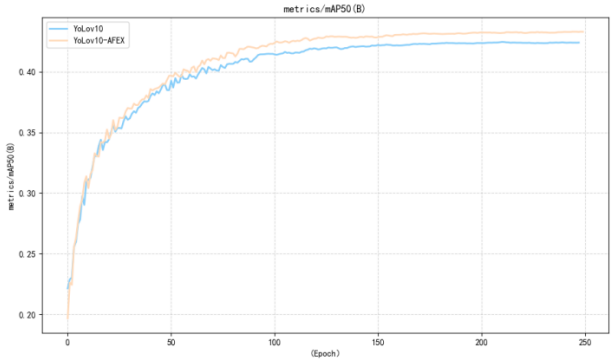

Figure 6. Comparison chart of mAP during training

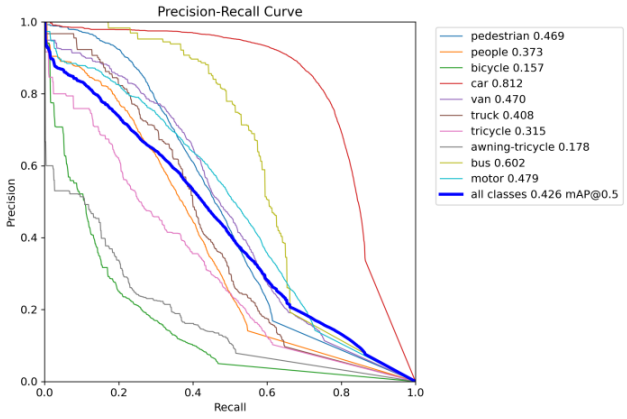

Figure 7. Average precision of each label in YOLOv10

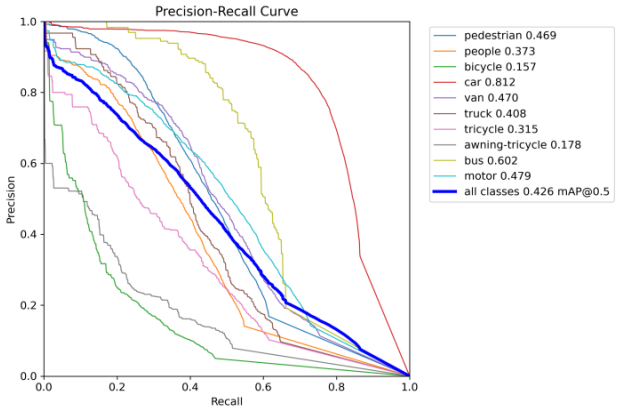

Figure 8. Average precision of each label in APFU-YOLOv10

The comparison of mAP during the training process before and after model improvement is shown in Figure 6. The average precision of the models before and after improvement on the test set is illustrated in Figures 7 and 8, respectively.

The detection performance for the pedestrian category improved by 0.9 percentage points; the categories bicycle and van, which were prone to confusion in the original model, improved by 2.5 and 2.0 percentage points, respectively; tricycle and motorcycle both saw improvements of 1.1 percentage points. The relatively poor performance of the original YOLOv10 model on bicycle and van categories is attributed to the limited receptive field, which leads to loss of regional features. The improved model adaptively adjusts the pooling methods based on the input features to better fuse multi-scale information.

D. Ablation Study

The proposed enhancements extend the YOLOv10 baseline architecture. Systematic ablation experiments were conducted to evaluate the impact of each module, with rigorous measurements of detection precision and computational overhead. These empirical findings are summarized in Table II.

TABLE II. RESULTS OF ABLATION EXPERIMENT

Model	*AFPB*	*EFA*	*Pram/M*	*FLOPs/G*	*FPS/f/s*	*mAP/%*
Ⅰ			7.22	21.4	128	42.6
Ⅱ	✓		7.24	21.5	127	43.0
Ⅲ		✓	7.68	21.8	125	42.9
Ⅳ	✓	✓	7.8	21.8	125	43.5

Experiment I: The baseline model was used as the control group in the ablation study. This model features a compact architecture and achieves a fast inference speed of 128 FPS; however, there remains room for improvement in detection accuracy under complex environments, with an mAP of 42.6%.

Experiment II: The false positive suppression module AFPB was introduced into the baseline model. AFPB effectively mitigates false detections in scenarios with multiple coexisting object categories, resulting in an mAP increase of 0.4 percentage points to 43.0%. The model parameters and computational complexity increased only slightly, with negligible impact on real-time performance, validating AFPB's effectiveness and lightweight advantage.

Experiment III: The efficient feature augmentation module EFA was incorporated individually to enhance the model's representational capability for small objects and complex scenes. Although this led to a relatively larger increase in parameters and computation, and the inference speed decreased to 125 FPS, the mAP improved by 0.3 percentage points to 42.9%, demonstrating the positive effect of the EFA module on detection accuracy in complex backgrounds.

Experiment IV: Both AFPB and EFA modules were integrated simultaneously. The model parameters increased to 7.80 million, with computation maintained at 21.8 GFLOPs, and inference speed remained unchanged. This configuration achieved the highest model accuracy in the ablation study, with mAP rising to 43.5%, an improvement of 0.9 percentage points over the baseline. This indicates that the combined use of these two modules significantly enhances model performance while maintaining computational efficiency.

E. Comparative Experiments with Different Detection Algorithms

To objectively assess the proposed algorithm's performance in UAV object detection, we conducted rigorous comparative experiments with conventional detection methods. All evaluations were performed under controlled experimental conditions, maintaining identical hardware configurations, software environments, and dataset partitions. Quantitative results are systematically compared in Table III.

TABLE III. THE RESULTS OF COMPARATIVE EXPERIMENTS ON DIFFERENT DETECTION ALGORITHMS

Algorithm	*P/M*	*FLOPs/G*	*AP/%*			*mAP@0.5%*
			Car	*van*	*bus*	
YOLOv5s	9.13	24.1	72.8	41.2	52.0	35
YOLOv8s	11.14	28.7	75.3	43.8	56.4	38.5
YOLOv10s	7.22	21.4	81.2	47.0	60.2	42.6
APFU-YOLOv10s	7.8	21.8	81.6	48.0	60.9	43.5

As shown in Table 3, the proposed improved model, APFU-YOLOv10s, achieves the highest detection accuracy with an mAP@0.5 of 43.5%, representing an improvement of 0.9 percentage points over the original YOLOv10s, while maintaining low computational overhead. Moreover, APFU-YOLOv10s retains a lightweight structure, with the number of parameters and computational complexity only slightly higher than that of the original YOLOv10s, and significantly lower than those of YOLOv5s and YOLOv8s. This demonstrates that the integration of AFPB and EFA in the APFU-YOLOv10s model effectively balances accuracy and

efficiency, offering better adaptability across various scenarios.

V. CONCLUSIONS

This study presents an enhanced object detection framework built upon the YOLOv10 architecture, designed to improve multi-scale feature learning and adaptive feature integration. First, efficient fusion of multi-level feature maps was achieved, which preserves the strong semantic information from low-resolution features while enhancing the detailed representation of high-resolution features. Second, by adaptively recalibrating feature weights, the model effectively strengthens the expression of informative features and suppresses irrelevant background interference. Finally, the network model constructed with the proposed modules significantly improves detection accuracy and robustness for objects of varying scales, while maintaining low computational overhead. Comprehensive experimental comparisons confirm the efficacy and advantages of our approach, highlighting its robust adaptability and practical viability in diverse real-world environments.

REFERENCES

[1] Rui Jiang, Jingxin Yang, Chengxi Gui, "Application of Deep Learning in Remote Sensing Image Processing," Wireless Interconnect Technology, vol. 19, no. 5, pp. 89–92, 2022.

[2] Chenxuan Li, Huiqi Xu, Kun Qian, et al., "A Review of Ship Target Detection Technology Based on Deep Learning," Journal of Ordnance Equipment Engineering, vol. 42, no. 12, pp. 57–63, 2021.

[3] S. Ren, K. He, R. Girshick, et al., "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 39, no. 6, pp. 1137–1149, 2017.

[4] Yanhua Shao, Duo Zhang, Hongyu Chu, et al., "Review of YOLO-Based Object Detection Using Deep Learning," Journal of Electronics & Information Technology, vol. 44, no. 10, pp. 3697–3708, 2022.

[5] Yecai Guo, Jingdong Sun, Amitave Saha, "An Improved YOLOv5-Based Object Detection Algorithm for Aerial Images," Journal of System Simulation, vol. 37, no. 2, pp. 551–562, 2025.

[6] Linglong Qi, Jianling Gao, "Small Object Detection Based on Improved YOLOv7," Computer Engineering, vol. 49, no. 1, pp. 41–48, 2023.

[7] Wei Pan, Chao Wei, Chunyu Qian, et al., "An Improved YOLOv8s Model for Small Object Detection from UAV Perspective," Computer Engineering and Applications, vol. 60, no. 9, pp. 142-150, 2024.

[8] Peng Li, Pengfei Li, Ruijie Yin, et al., "An Improved YOLOv10s-Based Method for UAV Target Detection and Recognition," Journal of Geomatics Science and Technology, vol. 41, no. 2, pp. 204-211, 2025.

[9] Kaijun Wu, Zhuo Pu, "A UAV Perspective Target Detection Algorithm Based on Feature Information Supplement and Enhancement," Journal of Beihang University, pp. 1–13, 2025.

[10] Qin Zhang, Weian Guo, "A Survey on Small Object Detection Algorithms Based on Deep Learning," Computer Applications and Research, pp. 1–14, 2025.

[11] Weifeng Gao, Yuxuan Yi, Lingling Huang, et al., "An Efficient Small Object Detection Algorithm for UAV Aerial Imagery," Control and Decision, pp. 1-9, 2025.

[12] Guokai Fu, "Research on Remaining Useful Life Prediction Model for Bearings Based on Convolutional Neural Networks," Science and Technology Innovation and Application, vol. 15, no. 17, pp. 64-67, 71, 2025.