YOLOXpress: A lightweight real-time unmanned aerial vehicle detection algorithm

Nguyen Tien Tai, Bui DucThang, Nguyen Ngoc Hung*

Institute of Control Engineering, Le Quy Don Technical University, Hanoi, Vietnam

*Corresponding author E-mail: hungnn@lqdtu.edu.vn

(Received 14 October 2024; Final version received 24 January 2025; Accepted 24 February 2025)

Abstract

The widespread use of drones has made drone detection a critical factor in various fields, particularly in security and defense. However, this task presents unique challenges due to the high speed, small size, and ability of drones to blend into their surroundings, which can hinder detection effectiveness. This paper introduces enhancements to the You Only Look Once (YOLO)-v8 model to improve real-time drone detection capabilities, especially when deployed on resource-constrained devices. We propose an improved model called YOLOXpress, which optimizes both processing speed and model size while maintaining an acceptable level of accuracy. By replacing the Cross-Stage Feature Fusion modules in the Backbone and Neck with Re-parameterization Convolution and RepC3 modules, we significantly reduced the number of computations, achieving a 12.25% increase in processing speed (frames per second) and a 69.96% reduction in model size. Although there was a 6% decrease in average accuracy compared to the original YOLO-v8 model, YOLOXpress remained effective for real-time drone detection. Experiments conducted on the TIB-Net dataset confirmed that this model is highly suitable for deployment on resource-limited devices, such as compact embedded systems.

Keywords: Drone Detection, Deep Learning, Real-Time Processing, Unmanned Aerial Vehicle, YOLO-v8

1. Introduction

The application of scientific and technological advancements in drone production is becoming increasingly widespread. Along with its exceptional advantages, drone production also comes with unforeseeable consequences. Small drones have become more prevalent in recent years, with various models performing various tasks. This directly threatens the security of many countries, as drones can be used for espionage, surveillance, and suicide missions, often equipped with weapons to target key locations with deliberate intent, thus forming new and unconventional methods of warfare (Al-lQubaydhi et al., 2024). To counter the threats posed by drones, developing and implementing anti-drone systems has become an urgent priority in modern defense and security.

Detecting drones and providing early warnings have received considerable attention from various research groups and have been extensively studied. In recent years, many studies have utilized deep-learning models for drone detection. Research employing computer vision and deep learning models, such as You Only Look Once (YOLO)-v3 (Alsanad et al., 2022), YOLO-v5 (Lv et al., 2022), and YOLO-v8 (Kim et al., 2023), has yielded promising results.

However, most present anti-drone devices face limitations, such as fixed installation requirements, large size, easy detectability, and difficulty in deployment in areas with space constraints. These limitations directly affect and reduce the effectiveness of the devices. Therefore, developing new deep-learning models with smaller sizes, real-time processing speeds, and higher accuracy for drone detection is essential. Developing such models would help reduce hardware resource usage when designing new anti-drone devices, thereby addressing the issue of device size limitations.

This paper is organized as follows: Section 1 introduces the research problem addressed in this study. Section 2 summarizes related works on unmanned aerial vehicle (UAV) detection; Section 3 begins with an overview of the methodology used in this research, followed by a presentation of the YOLOv8 network architecture, including detailed descriptions of its key modules. This section also presents the improved UAV target detection model and its architecture. Section 4 introduces the dataset and experimental setup, followed by ablation studies and comparison experiments using the publicly available TIB-Net dataset. It concludes with experiments conducted on a self-constructed dataset to validate the feasibility of the proposed method thoroughly. Finally, Section 5 summarizes the research findings and outlines potential future research directions.

2. Related Work

The task of object detection involves identifying objects within a specific frame. One commonly used method is leveraging convolutional neural networks (CNN) to extract and detect object features. Since the 2010s, as deep learning has advanced, the quality of object detection algorithms has continuously been upgraded and improved, with notable algorithms such as Region-Based CNN (RCNN) and faster RCNN. Although these networks have superior performance in terms of accuracy compared to classical algorithms, their complex structures hinder the models from achieving speeds equivalent to real-time processing. This is a critical requirement for real-world applications in object detection. Therefore, many studies have focused on creating models that balance speed and accuracy to enable wide practical implementation (Lee et al., 2019; Zhai et al., 2023; Zhu et al., 2021).

At present, the YOLO series of models has effectively addressed this problem. The YOLO series has undergone nine iterations of improvement, with several minor versions showing superior performance in both speed and accuracy (Terven et al., 2023). These models are widely applied across various fields, including medicine, transportation, industry, and UAV detection and warning systems. Research groups have conducted numerous studies on applying YOLO for UAV detection. Some studies have shown promising results, such as PaddlePaddle (PP)-YOLO (Long et al., 2020), an object detection method based on YOLOv3, optimized and improved to balance performance and efficiency. PP-YOLO employs existing techniques to improve object detection accuracy without increasing the number of model parameters and computations. With a mean average precision (mAP) performance of 45.2% and frames per second (FPS) speed of 72.9, PP-YOLO surpasses existing detectors, such as EfficientDet and YOLOv4. For Mob-YOLO (Liu et al., 2022), the authors proposed a lightweight model, an object detection method for UAVs. Based on the high-performance YOLOv4 model, MobileNetv2 (Sandler et al., 2018), a lightweight CNN, is used to replace the original CSPDarknet53 (Bochkovskiy et al., 2020) architecture of YOLOv4. This modification reduces the model size and simplifies computation, resulting in a significant increase in processing speed. Since 2023, improved models derived from YOLOv8 have been actively developed for UAV detection applications. For example, a study by Yılmaz & Oruç (2024) improved YOLOv8's performance in low-light environments by incorporating data augmentation techniques for brightness and color adjustment. It also enhanced feature extraction layers to optimize detection accuracy and speed, making the model more effective for real-time drone monitoring in challenging lighting conditions. Similarly, a study by Zamri et al. (2024) integrated attention modules and contextual learning to optimize UAV detection in aerial surveillance, improving performance in high-resolution video feeds and complex detection scenarios, such as identifying UAVs at high altitudes or in cluttered environments.

Today's primary challenges in applying deep learning to anti-drone systems are model size and real-time processing speed. A practical system, with a compact size and the ability to be installed in locations with limited space, requires simple and efficient hardware. Therefore, the hardware resources of anti-drone systems must be optimized. Although YOLO-v8 has been significantly improved in terms of performance and object detection capabilities, one notable drawback when deploying it on embedded devices is its slower processing speed compared to previous versions, especially when applied to devices with limited computational resources. Therefore, while YOLO-v8 brings substantial upgrades in object detection performance, it still faces limitations in terms of speed when deployed on embedded devices with constrained hardware configurations. This paper proposes a solution that improves computational speed while reducing model size. The approach discussed in this paper aims to minimize the number of computations, enhancing processing speed without compromising the accuracy necessary for object detection. To achieve this objective, a re-parameterization method (Wang et al., 2023) is employed. According to the study, the re-parameterization method integrates computational components into a single inference step. This method transforms a model with a complex structure during training into a significantly simpler structure when deployed on hardware devices, thereby increasing processing speed. With the observations and evaluations mentioned above, a new YOLOXpress model has been developed to address the limitations of YOLO-v8 in drone detection tasks. This paper presents the re-parameterization method used to create YOLOXpress based on the YOLO-v8 framework.

Several layers in the YOLO-v8 architecture are replaced to create YOLOXpress with features better suited to the task's requirements. Through experimentation, the model has achieved notable results and demonstrated advantages over YOLO-v8, such as being smaller, deployable on more cost-effective and smaller devices, and ensuring greater usability, flexibility, and the ability to be deployed in various locations, terrains, and conditions. Furthermore, the model strikes a balance between speed and accuracy, processing faster than YOLO-v8 while maintaining an acceptable level of accuracy.

3. Method

3.1. YOLO-v8 Architecture

YOLO-v8 is an upgraded version of the YOLO model series, designed to enhance speed and accuracy in real-time object detection tasks. The architecture of YOLO-v8 comprises three main components: Backbone, Neck, and Head. The Backbone extracts key features from images through a CNN network. The Neck utilizes techniques such as the feature pyramid network (FPN) and path aggregation network (PAN) to combine multiple-level features, improving object detection capabilities, particularly for small objects. The Head predicts bounding boxes and labels, with Non-Maximum Suppression reducing prediction overlap.

- Backbone: This component is responsible for extracting features from the input image. The Backbone typically uses deep CNN to learn low-level and high-level features from the image.
- Neck: This section enhances the features extracted by the Backbone. The Neck usually includes layers such as FPN or PAN to combine and further enrich the features.
- Head: The final part of the network is responsible for generating the final predictions. The Head predicts the bounding boxes and the classes of objects in the image.

YOLO-v8's loss function consists of object existence prediction, object classification, and accurate bounding box localization. Significant improvements such as Mosaic augmentation and other optimization techniques enhance YOLO-v8's learning and generalization abilities. However, for specific tasks, such as real-time UAV detection, YOLO-v8 reveals certain limitations, particularly in detecting small objects, failing to fully meet real-time requirements on embedded devices. This limitation arises from the computational inefficiencies of the Cross-Stage Feature Fusion (C2f) block, which remains cumbersome and suboptimal in terms of time efficiency. The architecture of C2f is shown in Fig. 1.

A layer with four characteristic parameters (*w, h*, C_m, C_{out}, *K*) was considered, where:

h denotes the height of the feature map,

w denotes the width of the feature map,

C_m represents the depth of the input feature map (with $C_{out} \geq C_m$),

C_{out} represents the depth of the output feature map,

and *K* represents the kernel size of the convolutional layer.

The computational cost of C2f was calculated by Eq. (1) (Wei et al., 2021):

$$Cost_{C2f} = Cost_{Conv1} + n * Cost_{BottleNeck} + Cost_{Conv2}$$

$$= h*w*C_{in}*C_{out}*K_1^2 + n*\begin{pmatrix} h*w*\frac{C_{out}}{2}*\frac{C_{out}}{4}+ \\ h*w*\frac{C_{out}}{2}*\frac{C_{out}}{4} \end{pmatrix}*$$

$$K_2^2 + h*w*(2+n)*\frac{C_{out}}{2}*C_{out}*K_2^2 \qquad (1)$$

where *n* is a parameter of C2f, representing the number of times the BottleNeck block is repeated, with $n \geq 1$, K_1 and K_2 are the sizes of the two standard CBS layers, where $K_1 = 1$ and $K_2 = 3$, respectively).

Assuming $C_m = 3$, $C_{out} = 32$, $h = 640$, $w = 640$ (standard input image size), the computational cost is given by $Cost_{C2f} = 196758400 + n*4718592000 \geq 6645350400$ (*parameters*).

Thus, even with the input image size and the number of iterations $n = 1$, the amount of computation required by C2f is still relatively large.

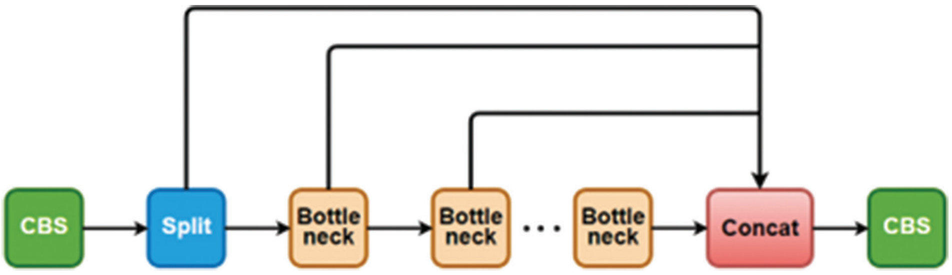

Fig. 1. Architecture of cross-stage feature fusion in YOLO-v8. Redrawn based on the original design by Zhai et al. (2023), with some adjustments to suit our study
Abbreviation: CBS: Convolution, batch normalization, and SiLU activation functions

3.2. YOLO-v8 Improvement for Real-time UAV Detection

For the real-time UAV detection task, the model must meet several requirements; it should be lightweight, suitable for embedded devices, and capable of accurately identifying small objects in real-time.

Based on the proposals and results from previous studies (Howard et al., 2017; Sandler et al., 2018; Terven et al., 2023), to reduce the model size, it is necessary to decrease the number of computations in the convolutional layers. Therefore, the C2f module was replaced with a simpler one.

Several challenges must be addressed regarding the small object detection problem: no object should be missed, and the object must be distinguishable from the background. Studies have proved that a 3×3 convolutional layer is effective in gathering local information while requiring lower computational costs compared to larger convolutional layers (Dosovitskiy, 2020; Szegedy et al., 2015; Szegedy et al., 2016). At present, the main approaches for small object detection rely on the Vision Transformer (ViT) architecture (Wu & Dong, 2023; Zhai et al., 2023; Zhu et al., 2021),(Dosovitskiy, 2020). This study showed that although ViT offers significant advantages in capturing global information, its cost is excessively high due to the use of the Self-Attention mechanism.

Therefore, this paper proposed an architecture block primarily based on 3 × 3 and 1 × 1 convolutional layers to retain the same level of information as the C2f block in YOLO-v8, but with reduced model size. The C2f layer in the Backbone was replaced with Re-parameterization Convolution (RepConv) to create a lighter Backbone while ensuring sufficient information is provided to the Neck. Subsequently, the C2f layer in the Neck was replaced with RepC3 to reduce computational complexity during feature extraction and aggregation while retaining essential object-related information.

3.2.1. Re-parameterization convolution

As presented above, we proposed an architectural block called RepConv in this section. The architecture of RepConv is shown in Fig. 2.

To compare RepConv with the C2f block, a layer with four characteristic parameters was considered (w, h, C_m, C_{out}, K).

The computational cost with RepConv was calculated by Eq. (2) (Wei et al., 2021):

$$Cost_{RepConv} = h*w*C_{in}*C_{out}*K_1^2 + h*w*C_{in} *C_{out}*K_2^2 \qquad (2)$$

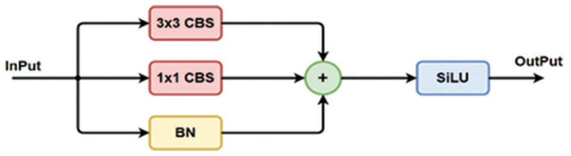

Fig. 2. Architecture Re-parameterization Convolution. Redrawn based on the original design by Zhai et al. (2023), with some adjustments to suit our study Abbreviation: BN: Batch normalization; CBS: Convolution, batch normalization, and SiLU activation functions; SiLU: Sigmoid linear unit

$$Cost_{RepConv} = 10*h*w*C_{in}*C_{out} \qquad (3)$$

On the other hand, Eq. (IV) is as follows:

$$Cost_{C2f} = h*w*C_{in}*C_{out} + 9*n*h*w*C_{out}*\frac{C_{out}}{2} + 9*h*w*(2+n)*\frac{C_{out}^2}{2} \geq h*w*C_{in}*C_{out} + 9*n*h* w*C_{out}*\frac{C_{out}}{2} + 27*h*w*\frac{C_{out}^2}{2} > 10*h*w*C_{out}^2 > 10*h*w*C_{in}*C_{out} \qquad (4)$$

Hence, $Cost_{RepConv} < Cost_{C2f}$

As presented in Section 3.2, the 3 × 3 convolutional layer captures local information in a smaller spatial area compared to larger convolutional layers, such as 5 × 5 or 7 × 7. In the case of the drone detection task, the object typically occupies a very small portion of the entire image space. Therefore, using a convolutional layer that observes a large spatial area made it challenging to detect the object's features.

In addition, including a 1 × 1 branch to retain the original information of the object, which is then aggregated, helps highlight the key features extracted by the 3 × 3 layer.

This demonstrates that RepConv performs well as expected, with the target object standing out relative to its surrounding area. RepConv is capable of replacing C2f in the Backbone to meet real-time requirements.

3.2.2. RepC3

As demonstrated in Section 3.2 regarding the effectiveness of RepConv and its comparison with C2f, the BottleNeck blocks in C2f were proposed to be replaced with RepConv blocks to improve the speed of the model's Neck. The architecture of RepC3 is shown in Fig. 3.

Computational cost with RepC3w as calculated using the following equation:

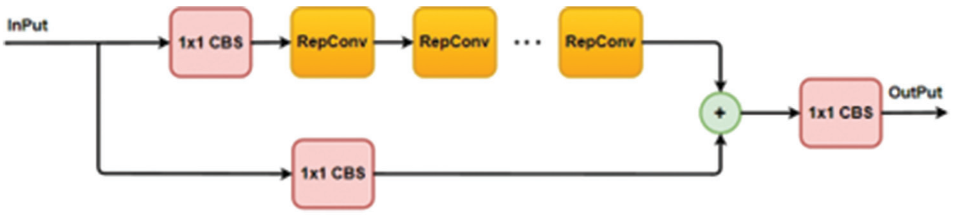

Fig. 3. Architecture of RepC3
Abbreviation: CBS: Convolution, batch normalization, and SiLU activation functions;
RepConv: Re-parameterization Convolution

$$Cost_{RepC3} = Cost_{Conv1} + n * Cost_{RepConv} + Cost_{Conv2} = h * w * C_{in} * C_{out} + n * 2.5 * h * w * \frac{C_{out}^{2}}{2} = h * w * C_{in} * C_{out} + (n * 2.5 + 0.5) * h * w * C_{out}^{2} \qquad (5)$$

From Eqs. (4) and (5), we can see: $Cost_{RepC3} < Cost_{C2f}$

Two modules, RepConv and RepC3, were developed based on the aforementioned theoretical frameworks and reasoning. Both architectures achieved the dual objectives of minimizing computational overhead while preserving the informational fidelity of target objects. These two architectural blocks were then integrated to replace the Backbone of the original YOLO-v8 model, completing the YOLOXpress model.

3.3. Architecture YOLOXpress

Based on the theoretical foundation presented in Sections 3.1 and 3.2, the C2f blocks were replaced with RepConv and RepC3, respectively, as shown in Fig. 4.

The architecture in Fig. 4 primarily illustrates that the C2f blocks are the main components being replaced. We also modified the model's upscaling method by employing a convolutional layer instead of interpolation, which was used in the original architecture. Techniques such as anchor-free design, Feature Pyramid architecture, CIoU loss, DFL, and BCE remained unchanged.

4. Experiment and Results

This paper employed the UAV dataset, originally used to train the TIB-net model (Sun et al., 2020), for training the YOLOXpress model. The results obtained from training were then utilized to evaluate the model's performance, conduct ablation experiments, and compare them with other models.

4.1. Dataset

The TIB-Net UAV dataset contains 2,850 images, capturing various types of UAVs, including multi-rotor UAVs and fixed-wing UAVs. These images were collected using a ground-based camera 500 m from the airborne UAVs, with a 1920×1080 pixels resolution. The scenery in the images includes low-altitude views, such as the sky, trees, and buildings, recorded at different times of the day and under various weather conditions. Analysis reveals that the UAVs occupy <1% of the area in each image.

4.2. Setup and Training Network

In this experiment, data from TIB-net were utilized to train the YOLO-v8 and YOLOXpress models. The experiment was performed on two pieces of hardware. The model training phase employed the Google Colab platform with a Tesla V100 GPU. After completing the training, the model was deployed on Jetson Orin Nano hardware. The NVIDIA Jetson Orin Nano 8GB Developer Kit was used for artificial intelligence processing applications, featuring a 6-core Arm® Cortex®-A78AE v8.2 64-bit CPU with 1.5MB L2 + 4MB L3 cache and an NVIDIA Ampere architecture GPU with 1024 CUDA cores and 32 Tensor cores. This configuration provides AI processing power up to 80 times greater than its predecessor, the Jetson Nano.

4.2.1. Loss function setting

The loss function of the YOLOXpress model is presented in Eq. (vi). It retains the same structure as the YOLOv8 loss functions, consisting of three components: rectangular box loss ($Loss_{Box}$), distribution focal loss ($Loss_{dfl}$), and classification loss ($Loss_{cls}$).

$$Loss = a * Loss_{box} + b * Loss_{dfl} + c * Loss_{cls} \qquad (6)$$

In this case, *a*, *b*, and *c* each represent the weighted proportion of the corresponding loss function in the overall loss function. In this experiment, the three weights were set as $a = 7.5$, $b = 1.5$, and $c = 0.5$.

4.2.2. Network training

Before training the network, the data directory was prepared in the YOLO-v8 format, which includes two folders: "images" and "labels." A batch size of 16

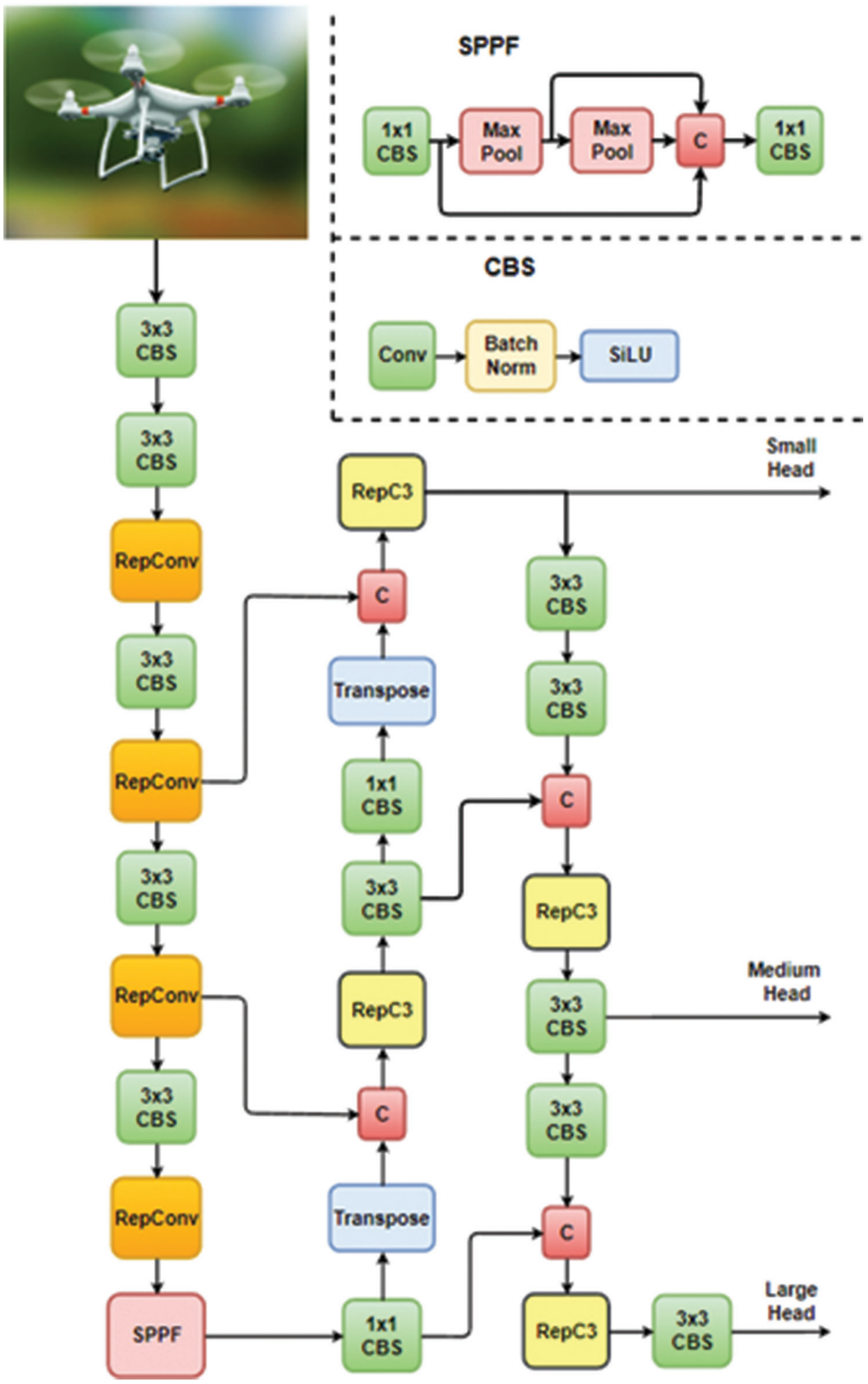

Fig. 4. Architecture of the YOLOXpress model
Abbreviation: CBS: Convolution, batch normalization, and SiLU activation functions; RepConv: Re-parameterization Convolution; SiLU: Sigmoid linear unit; SPPF: Spatial Pyramid Pooling Fusion

was chosen, and the model was trained for 250 epochs using an initial learning rate of 0.01. Table 1 describes the configuration parameters used during network training.

Table 1. Network training configuration

Parameter	**Values**
Epochs	250
Warm up epochs	10
Batch size	16
Image size	640×640
Initial learning rate	0.01
Final learning rate	0.01

4.3. Evaluation Metrics

To evaluate the model's quality, parameters, such as Precision (P), Recall (R), Average Precision (AP), mAP, the number of parameters, model size, and FPS were used.

The precision rate and recall were calculated using the following Eqs. (7) and (8), respectively:

$$P = \frac{TP}{\left(TP + FP\right)} \times 100\% \tag{7}$$

$$R = \frac{TP}{\left(TP + FN\right)} \times 100\% \tag{8}$$

True Positives (TP) represent the number of accurately detected objects, False Positives (FP) represent the number of non-target objects incorrectly detected as targets and False Negatives (FN) represent the number of targets not detected.

The AP and mAP were calculated using the following Eqs. (9) and (10), respectively:

$$AP = \int_0^1 p\left(r\right) d\left(r\right) \tag{9}$$

$$mAP = \frac{1}{N} \sum_{i=1}^{N} AP_i \tag{10}$$

where $p(r)$ is precision at recall r, and N denotes the total number of classes. In this paper, $N = 1$ corresponds to the task of UAV detection.

4.4. Ablation Experiments

In this section, the impact of each module replaced in the structure of the Model is clarified. Based on the TIB-net UAV dataset, replacement experiments were conducted on the original YOLO-v8 model, where modules in the Backbone and Neck were sequentially replaced according to a predetermined order. Four models were proposed for examination to analyze each module's impact. Model (a) is the standard YOLO-v8, Model (b) is the enhanced version with the RepC3 module replaced in the Neck, Model (c) is the enhanced version with the RepConv module replaced in the Backbone, and Model (d) is the enhanced version with both the RepC3 and RepConv modules replaced. The changes in these models were evaluated through the quantitative assessment of the parameters used to measure model performance, as presented in Table 2.

From the results in Table 2, the following observations can be made:

The model with the C2f module in the Backbone replaced by the RepConv module was proven effective in reducing the model size while maintaining performance. This is reflected in the data shown in Table 2, where the model sizes of (c) and (d) decreased by 11.17 MB and 12.44 MB, respectively, compared to model (a). Meanwhile, the P, R, and mAP parameters changed only slightly compared to model (a), with mAP decreasing by 0.244% for model (c) and 6.136% for model (d), and the changes in P and R being insignificant.

The reduction in the number of parameters, model size, and the number of computations in model (b) compared to model (a) demonstrates the effectiveness of the RepC3 module when replacing the C2f module in the Neck of model (a). The number of parameters decreased by 44.3%, the model size decreased by 80.8%, and GFLOPs decreased by 43.9%, while the P, R, and mAP parameters changed slightly compared to model (a). These findings prove that the RepC3 module effectively reduces computational load and model size while maintaining model performance.

The YOLOXpress model, which incorporated both the RepC3 and RepConv modules described in Section 3, showed a clear improvement in model size, GFLOPs, and parameters compared to the YOLO-v8n model, as indicated in Table 2. The performance metrics P, R, and mAP decreased slightly compared to model (a), but the differences were negligible. The difference between the models is shown in the feature map images in Fig. 5.

Four models were tested on Jetson Orin hardware, with measured results compared in Fig. 6. The findings demonstrate that the YOLOXpress model (modified by replacing the C2f module with the RepC3 and RepConv modules) achieved a balance between processing speed and accuracy. Specifically, compared to the original YOLOv8 model, YOLOXpress attained 93.99% accuracy while delivering a 12.25% increase in FPS and a 69.89% reduction in model size. These improvements made YOLOXpress highly suitable for deployment on low-configuration, compact-sized devices, which are ideal for applications demanding efficient object detection without compromising performance, even in hardware-constrained environments. The results underscore YOLOXpress's

Table 2. Results of the various ablation experiments

Component	**YOLO-v8 (a)**	**YOLO-v8 (RepC3) (b)**	**YOLO-v8 (RepConv) (c)**	**YOLOXpress (RepC3 and RepConv) (d)**
P	99.524	97.672	98.236	96.229
R	97.783	98.123	97.697	97.193
mAP	96.266	95.408	96.042	90.13
Parameter (million)	3.011	1.678	3.373	2.699
Model size	17.8	3.41	6.63	5.36
GFLOPs	8.2	4.6	9.3	7.6
Means of ACC	0.633	0.505	0.443	0.595
FPS	29.168	32.163	32.823	32.741

Abbreviations: ACC: Accuracy; AP: Average Precision; FPS: Frames Per Second; GFLOPs: Gigaflops mAP: Mean Average Precision; P: Precision; R: Recall; RepConv: Re-parameterization Convolution; YOLO: You Only Look Once.

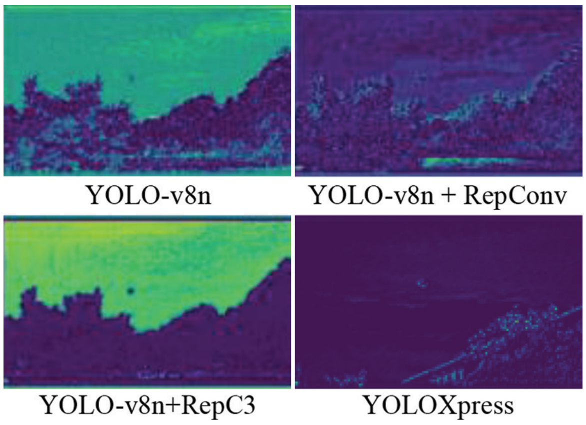

Fig. 5. Feature map extraction images of the test models and the YOLO-v8n model
Abbreviations: RepConv: Re-parameterization Convolution; YOLO: You Only Look Once

practical potential as a resource-efficient solution for real-world scenarios.

4.5. Comparative Experiment

To clarify the advantages of the YOLOXpress model, we compared it with the fastest models in the YOLO series, which are currently widely used on embedded devices (YOLOv8n, YOLOv6-s3.0 [Li et al., 2022], and YOLOv5n). The models were trained using a dataset we prepared, consisting of 8,022 images of various UAV with a resolution of 640 × 640 pixels. The results are presented in Table 3. The models selected for comparison are all official versions.

According to Table 3, it can be observed that: YOLOXpress achieved an FPS of 23.44, the highest among the models compared, demonstrating exceptional processing capability. This speed allows YOLOXpress to operate efficiently on embedded devices with limited computational resources.

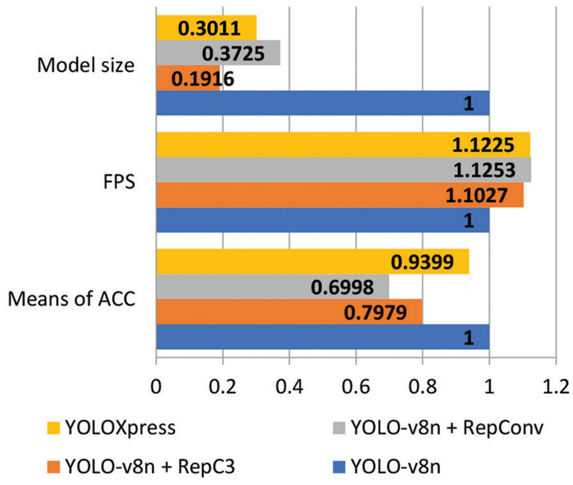

Fig. 6. Comparison chart between test models and the YOLO-v8n model (based on means of acc [accuracy], Frames Per Second [FPS], and model size)
Abbreviation: RepConv: Re-parameterization Convolution

Although YOLOv8n had the highest accuracy, its processing speed was only 18.02 FPS, significantly lower than that of YOLOXpress. This may affect applications that require real-time responsiveness. Other models, such as YOLOv5 (FPS = 20.05) and YOLOv6 (FPS = 19.52), also exhibited fast processing speeds but still fell short compared to YOLOXpress in scenarios demanding stringent real-time performance.

Although YOLOXpress did not achieve the highest accuracy (0.6243 compared to 0.6755 for YOLOv8n), it still maintained a strong performance in object detection with an mAP of 0.1636, which was close to YOLOv8n (mAP = 0.1750). This finding indicates that YOLOXpress strikes a good balance between accuracy and processing speed, which is crucial for applications that require both rapid processing and high reliability. While YOLOv6 and YOLOv5 demonstrated smaller model sizes and faster

Table 3. Comparison of experimental results

Parameter	YOLO -v8n	YOLO -v6	YOLO -v5	YOLO Xpress
Means of ACC	0.676	0.572	0.555	0.624
FPS	18.02	19.52	20.05	23.44
Model size (MB)				
Pytorch	5.9	49	5.0	5.1
Torchscript	11.9	16.5	10.1	10.1
Onnx	11.7	16.3	9.8	10.0
mAP (50-95)				
Pytorch	0.1750	0.1064	0.1588	0.1636
Torchscript	0.1732	0.1066	0.1584	0.1632
Onnx	0.1732	0.1066	0.1584	0.1632
Process time per image (ms/im)				
Pytorch	8.5	7.38	10.67	5.92
Torchscript	7.11	7.89	7.63	5.71
Onnx	137.24	178.43	146.65	119.82

Abbreviations: ACC: Accuracy; FPS: Frames per second.

processing speeds, they exhibited lower accuracy and mAP, which limits their effectiveness in scenarios that demand precise object detection.

With a model size of 5.1 MB, YOLOXpress delivered high performance and was also easily deployable on devices with limited memory, comparable to YOLOv5 (5.0 MB). This feature is crucial when deploying models on embedded devices or systems with constrained resources. While YOLOv8n and YOLOv6 showed larger model sizes (16 MB and 49 MB, respectively), the increased model size may require more powerful hardware and impact the ability to deploy on devices with limited memory and computational resources. YOLOXpress achieved the lowest processing time at 5.92 ms on the Pytorch platform, demonstrating its fast processing capability, which is well-suited for real-time detection applications.

Fig. 7 illustrates the real-world testing results in three scenarios: (i) Long-range detection under foggy weather conditions and complex background objects, (ii) long-range detection with an overcast sky, and (iii) detection in an environment with numerous complex objects. From the three provided images, we were able to assess the performance of the YOLOXpress model under these scenarios as follows:

Fig. 7A (long range, small target size, foggy conditions): The model successfully detected the drone despite low lighting and fog, which reduced contrast. The confidence score was 0.53, indicating that the model detected the drone with relatively low certainty due to the challenging conditions and small target size. These results demonstrate that the model can still perform acceptably under unfavorable conditions.

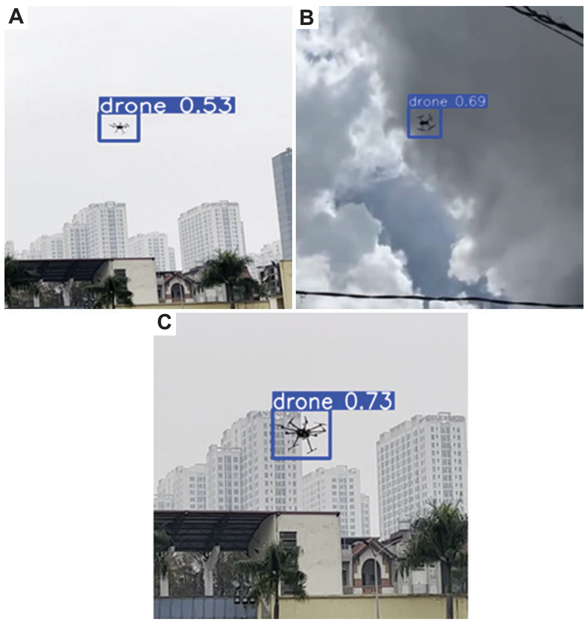

Fig. 7. Experimental results of the YOLOXpress model on the Jetson Orin embedded system under real-world conditions: (A) long-range detection with a small target size and foggy weather, (B) long-range detection with an overcast sky, and (C) detection in an environment with numerous complex objects

Fig. 7B (long-range, overcast sky): In the presence of thick clouds, the model detected the drone with higher confidence, achieving a score of 0.69. The detection performance improved compared to the first image, possibly due to the enhanced contrast between the drone and the overcast sky, making the target easier to identify. These findings suggest the model performs well even in complex sky conditions with fewer interfering objects.

Fig. 7C (environment with many interfering objects): The model achieved the highest confidence score of 0.73, successfully detecting the drone despite the presence of numerous background objects (buildings and trees). This finding demonstrates the model's robustness in handling complex environments with multiple potential sources of interference, particularly at close range. The successful detection in this scenario highlights YOLOXpress's ability to handle visually complex scenes.

In conclusion, the YOLOXpress model exhibited strong performance across various conditions, from unfavorable weather (fog) and overcast skies to environments with significant visual clutter. However, low lighting and small target sizes still impacted the model's confidence.

5. Conclusion

The YOLOXpress model proposed in this paper addresses the limitations of the YOLO-v8n model when deployed on low-end hardware devices, specifically for detecting small objects, particularly in UAV detection and alert systems. By prioritizing a small model size, fast processing speed, and maintaining an acceptable level of accuracy, YOLOXpress can be more easily deployed in resource-constrained environments. Specifically, the C2f module in the Backbone and the C2f module in the Neck can be replaced with the RepConv and RepC3 modules to reduce the number of computations while preserving the ability to extract object features. This modification reduces the model size without significantly affecting accuracy. Replacement and comparative experiments conducted on the TIB-Net dataset have provided specific metrics. Compared to the original model, the YOLOXpress model improved FPS and Model size by 12.25% and 69.96%, respectively. The parameters and computations were reduced by 10.36% and 7.32%, respectively.

In summary, the changes made to YOLOXpress compared to YOLO-v8 demonstrate that the model is suitable for deployment on low-end devices while ensuring real-time UAV detection. However, replacing the C2f module with the RepConv and RepC3 modules has resulted in a reduced accuracy compared to the original model. The average accuracy of the YOLOXpress model decreased by 6% compared to the original model. This reduction is minimal, and the accuracy remains within an acceptable range. Experiments on our custom-built dataset indicate that the recall rate decreases when more complex objects are in the background. Future work will improve accuracy and detection capability in complex background conditions.

References

Al-lQubaydhi, N., Alenezi, A., Alanazi, T., Senyor, A., Alanezi,N.,Alotaibi,B.,Alotaibi,M.,Razaque,A., & Hariri, S. (2024). Deep learning for unmanned aerial vehicles detection: A review. *Computer Science Review*, 51, 100614.

Alsanad, H.R., Sadik, A.Z., Ucan, O.N., Ilyas, M., & Bayat, O. (2022). YOLO-V3 based real-time drone detection algorithm. *Multimedia Tools and Applications*, 81(18), 26185–26198.

Bochkovskiy, A., Wang, C.Y., & Liao, H.Y.M. (2020). *Yolov4: Optimal Speed and Accuracy of Object Detection*. [arXiv Preprint] arXiv:2004.10934.

Dosovitskiy, A. (2020). *An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale*. [arXiv Preprint] arXiv:2010.11929.

Howard, A.G., Zhu, M., Chen, B., Kalenichenko, D., Wang,W., Weyand,T., Andreetto,M., & Adam,H. (2017). *MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications*. [arXiv Preprint] arXiv:1704.04861, p. 126.

Kim, J.H., Kim, N., & Won, C.S. (2023). High-Speed Drone Detection Based on Yolo-v8. In: *ICASSP 2023–2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*.

Lee, Y., Hwang, J.W., Lee, S., Bae, Y., & Park, J. (2019). An Energy and GPU-Computation Efficient Backbone Network for Real-Time Object Detection. In: 2019 *IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*.

Li, C., Li, L., Jiang, H., Weng, K., Geng, Y., Li, L., Ke, Z., Li, Q., Cheng, M., Nie, W., Li, Y., Zhang, B., Liang, Y., Zhou, L., Xu, X., Chu, X., Wei, X., & Wei, X. (2022). YOLOv6: A Single-Stage Object Detection Framework for Industrial Applications [arxiv Preprint].

Liu, Y., Liu, D., Wang, B., & Chen, B. (2022). Mob-YOLO: A Lightweight UAV Object Detection Method. In: *2022 International Conference on Sensing, Measurement and Data Analytics in the era of Artificial Intelligence (ICSMD)*.

Long, X., Deng, K., Wang, G., Zhang, Y., Dang, Q., Gao, Y., Shen, H., Ren, J., Han, S., & Ding, E. (2020). *PP-YOLO: An Effective and Efficient Implementation of Object Detector*. [arXiv Preprint]arXiv:2007.12099.

Lv, Y., Ai, Z., Chen, M., Gong, X., Wang, Y., & Lu, Z. (2022). High-resolution drone detection based on background difference and SAG-Yolov5s. *Sensors (Basel)*, 22(15), 5825. https://doi.org/10.3390/s22155825

Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., & Chen, L.C. (2018). Mobilenetv2: Inverted Residuals and linear Bottlenecks. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition.*

Sun, H., Yang, J., Shen, J., Liang, D., Ning-Zhong, L., & Zhou, H. (2020). TIB-Net: Drone detection network with tiny iterative backbone. *IEEE Access*, 8, 130697–130707. https://doi.org/10.1109/ACCESS.2020.3009518

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., & Rabinovich, A. (2015). Going Deeper with Convolutions. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., & Wojna, Z. (2016). Rethinking the Inception Architecture for Computer Vision. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

Terven, J., Córdova-Esparza, D.M., & Romero-González, J.A. (2023). A comprehensive review of yolo architectures in computer vision: From yolov1 to yolov8 and yolo-nas. *Machine Learning and Knowledge Extraction*, 5(4), 1680–1716.

Wei, T., Tian, Y., & Chen, C.W. (2021). Rethinking Convolution: Towards an Optimal Efficiency. In: *Under Review as a Conference Paper at ICLR*.

Wu, T., & Dong, Y. (2023). YOLO-SE: Improved YOLOv8 for remote sensing object detection and recognition. *Applied Sciences*, 13(24), 12977.

Yılmaz, H.B., & Oruç, F. (2024). Drone detection performance evaluation via real experiments with additional synthetic darkness. *Gazi University Journal of Science Part A: Engineering and Innovation*, 11(3), 546–562. https://doi.org/10.54287/gujsa.1526979

Zamri, F.N.M., Gunawan, T.S., Yusoff, S.H., Alzahrani, A.A., Bramantoro, A., & Kartiwi, M. (2024). Enhanced small drone detection using optimized YOLOv8 with attention mechanisms. *IEEE Access*, 12, 90629–90643. https://doi.org/10.1109/ACCESS.2024.3420730

Zhai, X., Huang, Z., Li, T., Liu, H., & Wang, S. (2023). YOLO-Drone: An optimized YOLOv8 network for tiny UAV object detection. *Electronics*, 12(17), 3664. https://doi.org/10.3390/electronics12173664

Zhu, X., Lyu, S., Wang, X., & Zhao, Q. (2021). TPH-YOLOv5: Improved YOLOv5 Based on Transformer Prediction Head for Object Detection on Drone-captured Scenarios. In: *2021 IEEE/CVF International Conference on Computer Vision Workshops (ICCVW)*.

AUTHOR BIOGRAPHIES

Nguyen Tien Tai is currently a Master's student at Le Quy Don Technical University. He earned his Bachelor's degree in Electrical and Electronics Engineering from Le Quy Don Technical University, Hanoi, Vietnam, in 2019.

Bui Duc Thang is currently a senior student at Le Quy Don Technical University.

Nguyen Ngoc Hung is currently a lecturer at Le Quy Don Technical University. He graduated with a Bachelor's degree in Electrical and Electronics Engineering from Le Quy Don University in 2010 and earned his Ph.D. from the same institution in 2023. His research interests include image processing for thermal camera, digital signal processing, and intelligent controller designing.