Journal of Computer and Communications, 2025, 13(4), 311-323
https://www.scirp.org/journal/jcc
ISSN Online: 2327-5227
ISSN Print: 2327-5219

Forest Fire Recognition Algorithm Based on Improved RT-DETR

Da Mu, Yunfeng Shang, Xinlei Hou

Graduate Student Office, North China Institute of Science and Technology, Dongyanjiao, Beijing, China
Email: 1413594600@qq.com

How to cite this paper: Mu, D., Shang, Y.F. and Hou, X.L. (2025) Forest Fire Recognition Algorithm Based on Improved RT-DETR. *Journal of Computer and Communications*, **13**, 311-323.
https://doi.org/10.4236/jcc.2025.134019

Received: April 24, 2025
Accepted: April 27, 2025
Published: April 30, 2025

Abstract

In recent years, with the frequent occurrence of global forest fires, fire prevention and control technology has become crucial. The advancement of artificial intelligence technology has provided emerging technical means for forest fire prevention. The RT-DETR model, as a newly emerged large model in recent years, has broken through the NMS limitations of the YOLO series and has shown great potential in the field of image recognition. However, due to its high resource consumption, it is not easy to deploy on embedded devices. To address this issue, a lightweight RT-DETR model has been proposed, which uses MobileNetV4 to optimize the backbone network, reducing parameters, enhancing computational speed, and lowering resource consumption. At the same time, to maintain the performance of forest fire detection, a Hierarchical Resolution Attention mechanism (HLF) and Learnable LPE encoding have been designed to ensure that the model remains lightweight without compromising detection capabilities. This improvement offers a new approach for the application of RT-DETR in the field of forest fire prevention.

Keywords

Deep Learning, Forest Fire Prevention, Wildfire Detection, Lightweight

1. Introduction

In recent years, the frequency of fire incidents worldwide has been increasing, making forest fire prevention particularly urgent. Against this backdrop, forest fire detection, as a key component of forest fire prevention, has become a hot topic in research. YOLO, as a classic algorithm in the field of image processing, has won high praise from the academic community for its efficient processing speed and ease of use. However, YOLO series algorithms generally rely on the Non-Maximum Suppression (NMS) step, which has become a bottleneck for improving their

performance.

In this context, the emergence of RT-DETR [1] has completely subverted the traditional processing methods. It adopts an end-to-end Transformer architecture and, through continuous improvements, it outperforms previous Transformer architectures in terms of performance, processing speed, and model size. RT-DETR has thus become a rising star in the field of image recognition, with potential that can match or even surpass YOLO in certain aspects. Currently, research on applying RT-DETR to forest fire recognition is limited, indicating that the potential of RT-DETR in the field of forest fire prevention has not been fully explored. Looking forward, RT-DETR has a broad application prospect in forest fire prevention with infinite potential for development. The trained RT-DETR model needs to be deployed and run not only on cloud platforms but also efficiently on embedded or mobile devices. This is because cloud platforms are typically located in remote locations far from the fire scene, making it difficult to achieve real-time response and processing of fire situations. Therefore, deploying the model to onsite embedded devices is particularly crucial. However, a major limitation of embedded devices is their relatively limited computing resources, which makes it difficult to process large amounts of data quickly. To this end, it is necessary to lightweight the model to ensure it remains efficient on resource-constrained devices. Jin, L [2] and others have reduced the consumption of computing resources through reparameterization to facilitate model deployment. Li, J and others [3] have reduced the model's size by using MobileNetv3 instead of conventional convolution modules. Huang, J [4], Chen, G [5], and others have proposed using ghost and seNet modules to simplify the model. It is evident that ghostnet and mobilenet are very popular network structures, each accounting for 40%, and they are the focus of research on lightweight network structures now and in the future. In addition to lightweight networks, model compression can also be achieved through knowledge distillation, channel pruning, and other operations. Wang S and others [6] have proposed using channel-level sparsity to simplify the model by assigning channels a certain proportion factor and then eliminating channels with small factors through training. Zhou, M and others [7] have used semi-supervised knowledge distillation techniques to simplify the YOLO model's architecture, reducing its complexity and improving detection accuracy. Although these methods do make the network structure more lightweight and easier to deploy on embedded devices, they often lead to a decrease in model performance compared to the original structure. Therefore, an improved RT-DETR model has been designed, aiming to balance lightweight and detection effectiveness. By using the newly proposed MobileNetV4 network to slim down the model, lightweighting has been achieved. Then, by replacing the original sine and cosine encoding strategy with a learnable encoding strategy, the model can effectively capture pixel location information, leading to more precise processing and generation of encoded data. Finally, the HLF module has been introduced to deeply integrate features from different levels, including semantic and detailed information,

thereby further enhancing the model's detection capabilities.

2. Materials and Methods

2.1. Baseline RT-DETR Model

The RT-DETR model has set a precedent for the first real-time end-to-end object detection model. This object detection model, based on the Transformer architecture, aims to surpass YOLO and achieve more efficient and rapid real-time object recognition. RT-DETR fully leverages the potential of Transformers in the field of computer vision, elevating the performance of real-time object detection to new heights. Like DETR, RT-DETR consists of a backbone network and a hybrid encoder.

The backbone is the initial stage where input images are processed to extract features. This typically involves a convolutional neural network that reduces the spatial dimensions of the image while increasing depth, generating feature maps at various stages (S3, S4, S5).

The hybrid encoder comprises two main parts: Attention-based Intra-scale Feature Interaction (AIFI) and Cross-scale Feature Fusion (CCFF).

AIFI processes only the S5 feature map from the backbone. Since S5 represents the deepest layer, it contains the richest semantic information about complete objects and their context in the image—making it highly suitable for the Transformer-based attention to capture meaningful relationships between objects. Although S3 and S4 contain useful low-level features such as edges and object parts, applying attention to these layers would be computationally expensive and less efficient because they do not yet represent complete objects that need to be related to each other.

CCFF, based on CNN layers, is responsible for combining the S3, S4, and AIFI (S5) feature maps. Its task is to fuse features from different scales, ensuring that the final feature map contains information from a variety of resolutions. The core of CCFF is a special block called RepBlock, which uses a structure named RepConv. RepConv allows the network to switch between different forms of convolution operations during training and inference, thus improving efficiency without sacrificing performance.

Following this, uncertainty queries are performed, and after selecting the most confident queries, RT-DETR uses a Transformer decoder with multi-scale deformable attention to predict object locations and categories. The decoder architecture is designed for efficient processing of multi-scale feature maps while maintaining high precision.

2.2. Improved Model RT-DETR-HMI

To facilitate the deployment of the model on embedded devices for on-site real-time forest fire detection while maintaining its detection performance, several improvements have been made. Firstly, to address the issues of large parameter size and computational complexity in RT-DETR, the backbone network has been re-

placed with a lightweight MobileNetV4 structure. To counteract the decrease in detection accuracy after lightweighting, the encoding method has been improved to the LPE (Learning Potential Encoding) method, which allows the model to better learn feature information. Additionally, the SDI (Scale-Discrepancy Integration) module has been employed to enhance the integration of fire features at different resolutions. The improved structure is shown in **Figure 1**.

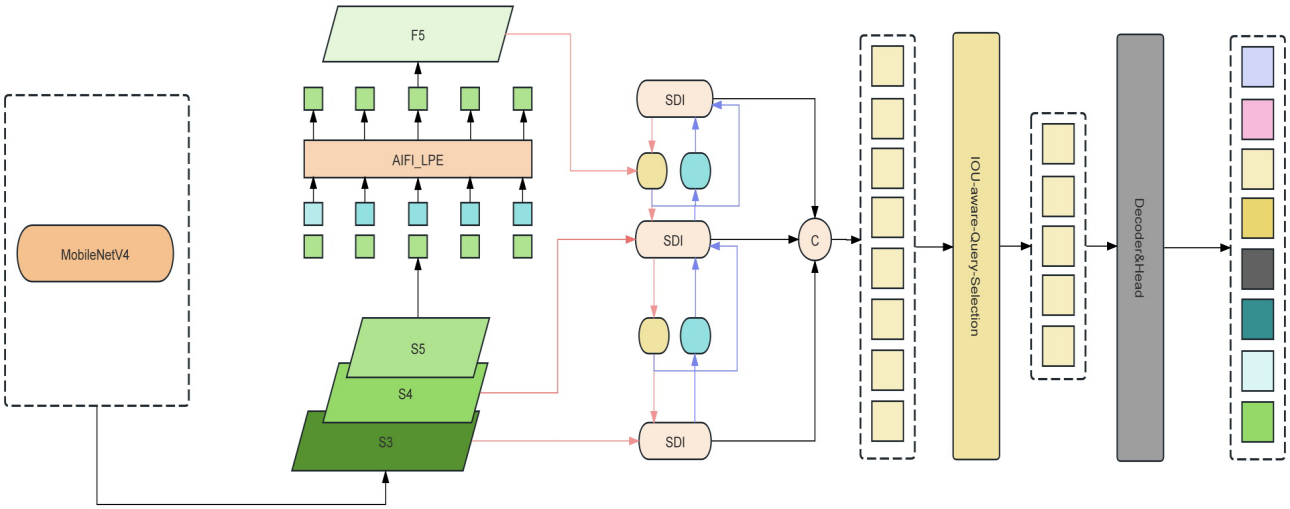

Figure 1. Structure of the improved RT-DETR model.

2.3. MobileNetV4 Lightweight Module

The latest generation of MobileNets, MobileNetV4 [8], presents a general and efficient architecture design tailored for mobile devices. At its core, the research team has introduced a search block known as Universal Inverted Bottleneck (UIB). This structure is unified and flexible, ingeniously integrating the Inverted Bottleneck (IB) [9], ConvNext [10], Feed-Forward Network (FFN), and an innovative Extra Depthwise (ExtraDW) [11] variant. In addition to UIB, the research team has also proposed Mobile MQA, an attention block specifically designed for mobile accelerators that can significantly enhance acceleration performance by 39%. Furthermore, to further improve the search efficiency of MNv4, the team has developed an optimized Neural Architecture Search (NAS) recipe.

(1) UIB

As shown in **Figure 2**, in the design of the inverted bottleneck, two optional Depthwise layers have been added. They are placed before the expansion layer and between the expansion layer and the projection layer. The inclusion of these layers is a key decision in the Neural Architecture Search (NAS) optimization process, leading to the creation of a new network architecture. Although this adjustment may seem simple, it ingeniously integrates multiple key network blocks, including the original Inverted Bottleneck (IB) block, the ConvNext block, and the Feed-Forward Network (FFN) block from ViT. Additionally, the UIB block introduces an innovative variant, the Extra Depthwise Inverted Bottleneck (ExtraDW) block.

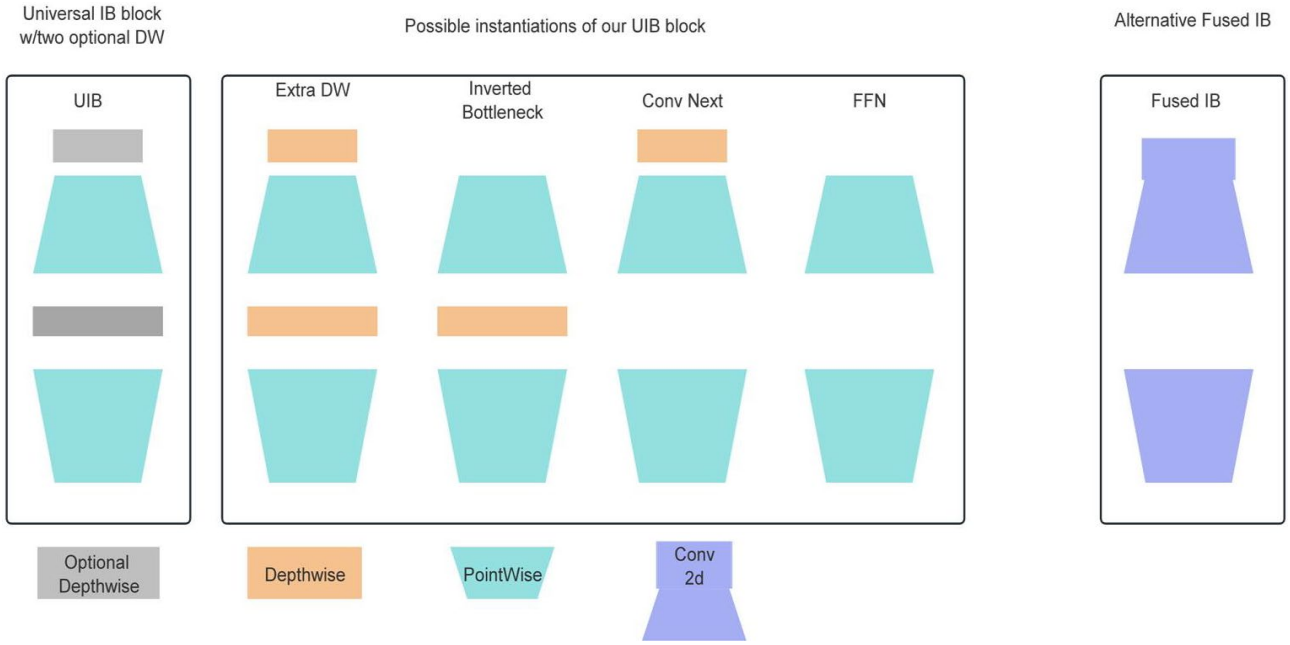

Figure 2. UIB and various IB combination structure modules.

The Inverted Bottleneck (IB) increases the model's capacity at the cost of increased computational expense by performing spatial mixing on the expanded feature activations. ConvNext achieves a less costly spatial mixing by performing the spatial mixing operation before feature expansion and allows for the use of larger convolution kernel sizes. ExtraDW is a new variant proposed in this article that enhances the network's depth and receptive field at a lower cost, combining the advantages of ConvNext and the Inverted Bottleneck (IB). The FFN consists of two stacked 1x1 pointwise convolutions (PW) layers, interspersed with activation layers and normalization layers. PW operations are among the most effective on accelerators, but their impact is most significant when used in conjunction with other network blocks.

(2) Mobile MQA

MQA significantly reduces the burden of memory access by implementing the sharing of keys and values across all attention heads. Building on this, a strategy of Spatial Reduction Attention (SRA) is further adopted, applying an asymmetric spatial downsampling technique to the keys and values to enhance computational efficiency. In the implementation details of MQA, a 3 × 3 depthwise separable convolution is used to replace the traditional average pooling step, effectively halving the spatial resolution of the keys and values. This dramatically reduces the inference time. The following is the mathematical expression for Mobile MQA:

$$MobileMQA(x) = Concat(attention_1 \cdots attentiuon_n)W^O \tag{1}$$

$$attention_j = \left(\frac{\left(XW^{Q_j}\right)\left(SR(X)W^k\right)^T}{\sqrt{d_k}}\right)\left(SR(X)W^k\right)^V \tag{2}$$

SR stands for spatial reduction, which involves asymmetric spatial downsampling of keys and values to further enhance efficiency.

2.4. AIFI-LPE Encoding

In the attention mechanism of transformers, to address the issue of ambiguous word vector positions in parallel operations, positional encoding is used to imbue each word vector with positional semantic information. This allows the attention mechanism to operate in parallel more effectively. The original positional encoding in the Transformer model uses a combination of sine and cosine functions [12], which is relatively fixed and may not be ideal for expressing complex positional information. Therefore, Localized Positional Encoding (LPE) is adopted.

LPE treats positional information as a parameter during the training process, continuously adjusting the encoding information in each training iteration. This allows for a more thorough representation of the positional relationships between each vector, thereby enhancing the effectiveness of the attention mechanism. Moreover, since LPE is trained on different datasets, it can adapt to various data distributions.

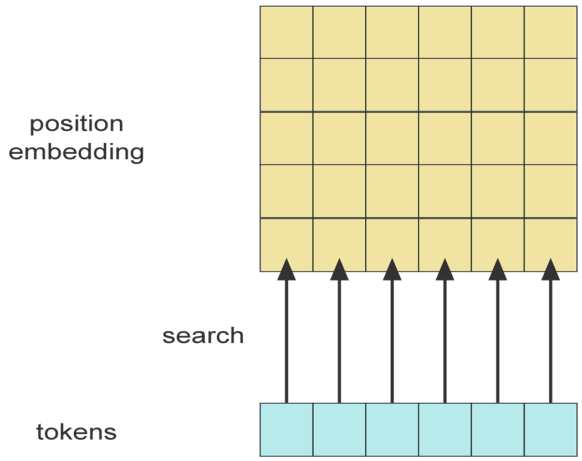

Figure 3. Schematic diagram of learnable location coding.

As shown in **Figure 3**, each element in the token sequence finds its corresponding positional encoding through a lookup in the embedding table. The positional encoding is then added to the element to obtain a vector representation that includes positional information. Subsequently, during training, the loss function is computed, and during the backpropagation process, the weight information in the position embedding is optimized. This results in more suitable positional information for each element, leading to better training outcomes for the attention mechanism.

2.5. HLF Module

In the basic RT-DETR module, the feature fusion module aims to reduce computational complexity and the consumption of computational resources. It overlooks

low-level detail information and prioritizes high-level semantic information, leading to insufficient fusion of high and low-level feature information. Therefore, the HLF module is proposed to enable more thorough fusion of high and low-level feature information. The schematic diagram of the module structure is shown in Figure 4.

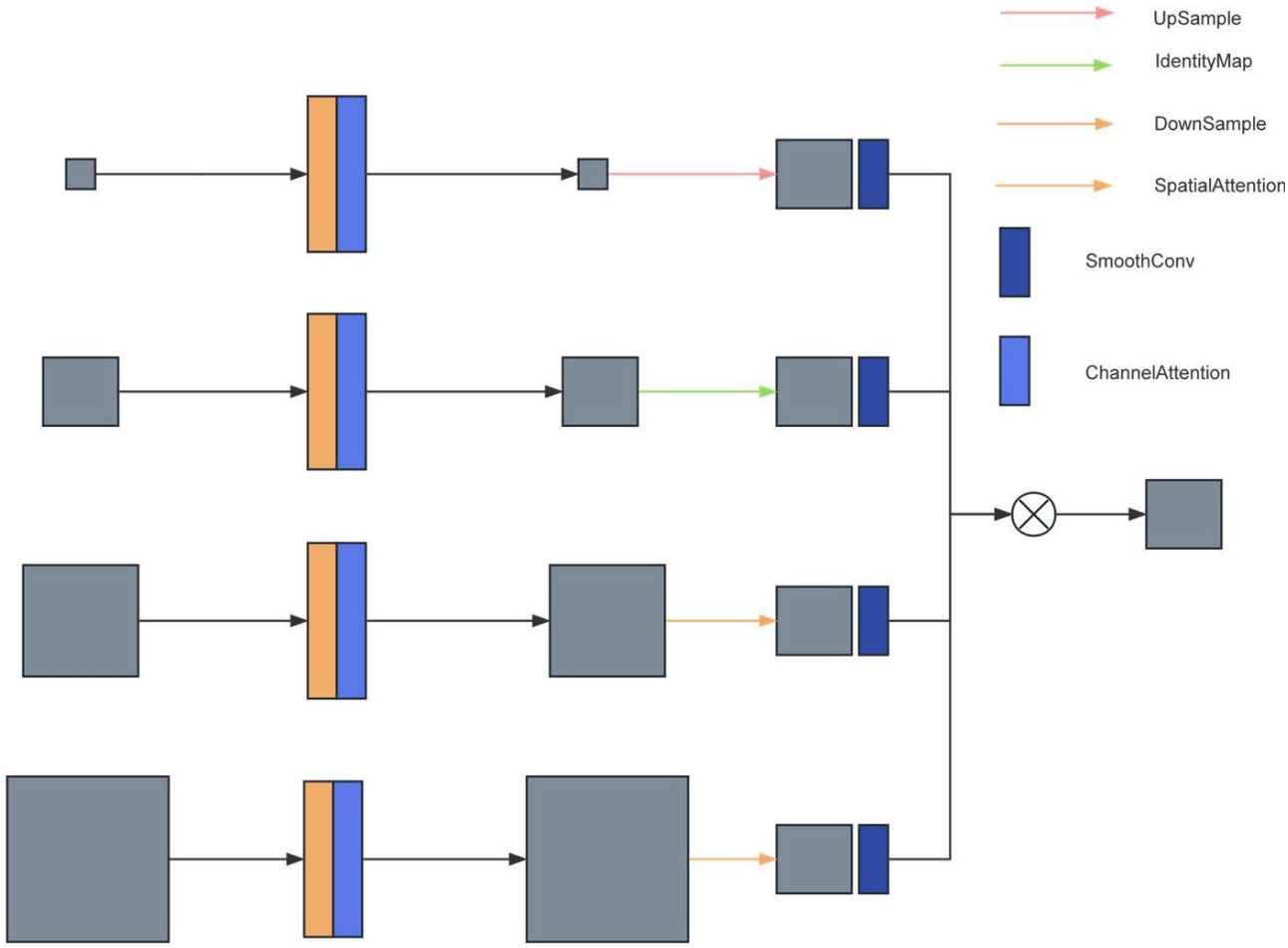

Figure 4. HLF module structure diagram.

The illustration shows the HLF (Hierarchical Level Fusion) module. Initially, the feature layers of various scales from the encoder input are taken as inputs and processed through channel attention mechanisms [13] and spatial attention mechanisms [14], respectively. The spatial attention mechanism focuses on identifying key regions within the feature maps, while the channel attention mechanism concentrates on selecting the primary channels within the feature maps. By utilizing these two attention mechanisms, the feature maps are able to fuse local spatial details with global channel information, resulting in a more expressive feature representation. This process is as shown in Equation 3:

$$f_i^1 = \varphi_i^c\left(\varphi_i^s\left(f_i^0\right)\right) \tag{3}$$

Formula 3 represents the feature integration process on, where f_i^0 denotes

the feature layer at the i-th scale that is input after passing through the encoder. φ_i^s represents the feature at the i-th layer after being processed by the spatial attention mechanism, and φ_i^c indicates the feature at the i-th layer after being processed by the channel attention mechanism.

Following this, feature scale adjustment is performed. If the size of the existing feature map is smaller than the target size, bilinear interpolation is used to upscale it to the target resolution. If the existing feature map is larger than the target size, adaptive average pooling is applied to downscale it to the target resolution. If the feature map's size already matches the target resolution, it is used directly. This process is shown in formula 4:

$$f_{ij}^3 \begin{cases} D\left(f_i^2,\left(H_i,W_i\right)\right) & j<i \\ I\left(f_i^2\right) & j=i \\ U\left(f_i^2,\left(H_i,W_i\right)\right) & j>i \end{cases} \tag{4}$$

Formula 4 represents the fusion of feature maps at different resolutions. D denotes downsampling operations when the image resolution is higher than the target image resolution. I indicates that the image is used directly when the image resolution is equal to the target image resolution. U represents upsampling operations using bilinear interpolation when the image resolution is lower than the target resolution, to restore it to the target resolution.

For each feature map that has been resized, a 3 × 3 convolution kernel is applied for smoothing. This 3 × 3 convolution effectively removes interference noise while preserving the image details, resulting in clearer feature maps.
This process is shown in Equation 5 as follows:

$$f_{ij}^5 = \theta_{ij}\left(f_{ij}^3\right) \tag{5}$$

where θij represents the parameters of the smooth convolution, and fij is the j-th smoothed feature map at the i-th level.

Finally, a Hadamard product [15] operation is applied to all the adjusted feature maps to generate the merged feature map. The Hadamard product is an effective method that combines two feature maps while retaining rich detail information.

3. Experimental Environment and Dataset

3.1. Experimental Environment

In this experiment, the training parameters are set as follows: the number of epochs is set to 100, image size is set to 640, batch size is set to 16, and the learning rate (lr) is set to 0.1. The optimizer chosen is AdamW, with a weight decay of 0.0001 and a global gradient clipping range of 0.1. The network was not initialized with pre-trained weights. Data augmentation includes random color distortion, expansion, cropping, flipping, and resizing. The linear warm-up steps are set to 10. The Hardware and Software Configuration of the experimental environment is shown in **Table 1** below.

Table 1. Environment configuration table.

environment	version
OS	Windows11
python	3.8.8
pytorch	2.20
GPU	NVIDA GEForce RTX 3060
CPU	12th Gen Intel Core i5-12400F

As shown in Table 1, The operating system used is Windows 11, with Python version 3.8.8. The PyTorch version is 2.20. For GPU, it's equipped with an NVIDA GEForce RTX 3060, and the CPU is the 12th Gen Intel Core i5-12400F.

3.2. Experimental Data

The dataset used in this experiment was created personally. The author provides the dataset at the following address: https://download.csdn.net/download/m0_64879847/88535127. To evaluate the effectiveness of the improved RT-DETR detection algorithm in the field of small target detection using unmanned aerial vehicle (UAV) aerial photography. The data was collected using a drone, The drone model is the DJI M300 RTK, operating at an altitude of 300 - 500 meters, and is equipped with an H20T payload camera. The video resolution is 1920 × 1080 at 30 frames per second. Data is transmitted to the backend server through CAN/4G/WiFi and other communication protocols. After frame extraction and data cleansing, an aerial forest fire image dataset is produced. This dataset encompasses records of forest fire incidents within the central region of the country over the past five years, spanning from 2020 to 2025 and covering multiple forested areas. The dataset is categorized into two classes: the flame dataset and the non-flame dataset. To enhance the completeness of the sample dataset, images of both small and large fire spots, as well as various environmental backgrounds under different seasonal conditions and daylight/darkness scenarios, have been collected, encompassing a range of forest vegetation coverage states. For the object detection task, 3699 images were selected as the training set, 800 images as the test set, and 500 images as the validation set. The resolution is 512 × 512 pixels.

4. Experimental Results and Analysis

4.1. Evaluation Metrics

The evaluation metrics used in this experiment are precision (P), recall®, mean Average Precision with an IOU threshold of 0.5 (mAP50), number of parameters (Parameter), and GFLOPS.

Precision refers to the proportion of samples that we predict as positive that are actually positive. Recall refers to the proportion of samples that are actually positive that we correctly predict as positive. mAP50 is an indicator that refers to the

calculation of Average Precision (AP) with an IoU threshold set to 0.5. This means that a prediction is considered accurate (*i.e.*, judged as a true positive) only when the IoU score between the predicted bounding box and the ground truth bounding box is not less than 0.5. The setting of the IoU threshold directly affects the calculation of AP, and mAP50 is a widely adopted evaluation standard because it effectively balances the trade-off between the accuracy of the bounding box localization and the detection criteria. The formulas for P, R, and mAP are as follows:

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{6}$$

$$\text{Pecall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{7}$$

$$\text{mAP} = \frac{\sum_{i=1}^{k} \text{AP}_i}{k} \tag{8}$$

TP stands for the number of positive samples that are predicted as positive, FP stands for the number of negative samples that are predicted as positive, and FN stands for the number of positive samples that are predicted as negative.

mAP represents the sum of the average precision values for all classes divided by the number of classes.

Here is the translation of your last sentence with the correct context:

mAP represents the sum of the average precision values for all classes, normalized by the number of classes.

4.2. Comparative Study

The results in Table 2 show that, when compared with the classic algorithms of the YOLO series through the improved RT-DETR, the enhanced algorithm demonstrates improvements in terms of precision (P), recall®, and mAP50. This indicates that RT-DETR, as a detection algorithm, can perform well in the field of forest fire prevention.

Table 2. Comparative experimental results.

model	P/%	R/%	mAP50/%	Params/M	GFLOPS
Yolov5	0.511	0.465	0.458	2508854	7.1
Yolov8	0.502	0.422	0.413	6092408	11.7
Yolov10	0.501	0.45	0.46	8836518	24.4
ours	0.516	0.458	0.491	11441624	39.6

4.3. Ablation Study

As shown in Table 3, The ablation study validates the performance improvement of the model with the addition of different modules. Firstly, by employing the lightweight module MobileNetv4 and through meticulous design of the network

Table 3. Results of ablation experiment.

model	P/%	R/%	mAP50/%	Params/M	GFLOPS
RT-DETR	0.502	0.421	0.441	32811576	108.0
+Mobilenetv4	0.512	0.437	0.446	11311576	39.5
+LPE	0.511	0.453	0.464	11413976	39.5
+HLM	0.516	0.448	0.472	11441624	39.6

architecture, such as the judicious selection of layer width (number of channels), stride, and layer stacking methods, it is possible to reduce the complexity of the model while maintaining accuracy. MobileNetV4 utilizes an improved inverted residual structure, which first increases the number of channels with pointwise convolutions, then applies depthwise convolutions, and finally uses pointwise convolutions to reduce the number of channels. This design aids in enhancing the network's representational capacity while maintaining its lightweight nature. Through the detailed design of the network architecture, the model's parameters and GFLOPS are reduced by approximately two-thirds. By employing Learnable Positional Encoding (LPE), positional encoding becomes a learnable form, allowing for specific adjustments of the parameters based on the dataset, thereby providing a more accurate description of the positional relationships between targets. This is beneficial for feature extraction, leading to an increase in both the R and mAP50 metrics by about 2 percentage points. Finally, the Hierarchical Local-Global Module (HLM) is added. This module further enhances the fusion and connection between high-level semantic features and low-level detail features, leading to a more comprehensive expression of target features. As a result, the model shows an almost one-percentage-point increase in both the P and mAP50 metrics. This confirms the superior performance of the improved model over the original model in the domain of forest fire detection.

RT-DETR ours

Figure 5. Comparison of visual results.

It can be clearly observed from Figure 5 that the optimized RT-DETR model excels in extracting flame features and fusing flame features at different scales, thereby significantly improving the recall rate and enhancing the precision of the model's detection. Particularly in the detection of small target flames, its detection capability has been significantly enhanced.

5. Conclusion

In response to the challenges of deploying the original RT-DETR model on embedded devices, the introduction of the MobileNetV4 lightweight structure has reduced the model parameters by 60%, significantly decreasing the resource consumption of the devices. Building on this, an innovative LPE encoding scheme and HLM feature fusion structure have been designed, effectively enhancing the detection performance of the improved RT-DETR, with an increase of 3.1 percentage points in mAP50. This improvement considers both the size of the model and its performance, demonstrating the good lightweight potential of the RT-DETR model. By lightweighting the RT-DETR model and optimizing the encoding method and feature fusion strategy, the model maintains detection performance while being more deployable on embedded devices. This provides a useful reference for the application of the RT-DETR model in the field of forest fire detection. However, its detection speed still lags significantly behind YOLO. Therefore, further optimization of the detection speed should be prioritized in the next phase of work to enhance the model's real-time capabilities.

Conflicts of Interest

The authors declare no conflicts of interest regarding the publication of this paper.

References

[1] Zhao, Y., Lv, W., Xu, S., Wei, J., Wang, G., Dang, Q., *et al.* (2024) DETRs Beat YOLOs on Real-Time Object Detection. 2024 *IEEE/CVF Conference on Computer Vision and Pattern Recognition* (*CVPR*), Seattle, 16-22 June 2024, 16965-16974. https://doi.org/10.1109/cvpr52733.2024.01605

[2] Jin, L., Yu, Y., Zhou, J., Bai, D., Lin, H. and Zhou, H. (2024) SWVR: A Lightweight Deep Learning Algorithm for Forest Fire Detection and Recognition. *Forests*, **15**, Article 204. https://doi.org/10.3390/f15010204

[3] Li, J., Tang, H., Li, X., Dou, H. and Li, R. (2023) LEF-YOLO: A Lightweight Method for Intelligent Detection of Four Extreme Wildfires Based on the YOLO Framework. *International Journal of Wildland Fire*, vol. 33. https://doi.org/10.1071/wf23044

[4] Huang, J., He, Z., Guan, Y. and Zhang, H. (2023) Real-Time Forest Fire Detection by Ensemble Lightweight YOLOX-L and Defogging Method. *Sensors*, **23**, Article 1894. https://doi.org/10.3390/s23041894

[5] Chen, G., Cheng, R., Lin, X., Jiao, W., Bai, D. and Lin, H. (2023) LMDFS: A Lightweight Model for Detecting Forest Fire Smoke in UAV Images Based on YOLOv7. *Remote Sensing*, **15**, Article 3790. https://doi.org/10.3390/rs15153790

[6] Wang, S., Zhao, J., Ta, N., Zhao, X., Xiao, M. and Wei, H. (2021) A Real-Time Deep Learning Forest Fire Monitoring Algorithm Based on an Improved Pruned + KD

Model. *Journal of Real-Time Image Processing*, **18**, 2319-2329. https://doi.org/10.1007/s11554-021-01124-9

[7] Zhou, M., Wu, L., Liu, S. and Li, J. (2023) UAV Forest Fire Detection Based on Lightweight YOLOv5 Model. *Multimedia Tools and Applications*, **83**, 61777-61788. https://doi.org/10.1007/s11042-023-15770-7

[8] Qin, D., Leichner, C., Delakis, M., Fornoni, M., Luo, S., Yang, F., *et al.* (2024) Mobilenetv4: Universal Models for the Mobile Ecosystem. In: Leonardis, A., Ricci, E., Roth, S., Russakovsky, O., Sattler, T. and Varol, G., Eds., *Computer Vision—ECCV* 2024, Springer, 78-96. https://doi.org/10.1007/978-3-031-73661-2_5

[9] Sandler, M., Howard, A., Zhu, M., Zhmoginov, A. and Chen, L. (2018) MobileNetV2: Inverted Residuals and Linear Bottlenecks. 2018 *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, Salt Lake City, 18-23 June 2018, 4510-4520. https://doi.org/10.1109/cvpr.2018.00474

[10] Liu, Z., Mao, H., Wu, C., Feichtenhofer, C., Darrell, T. and Xie, S. (2022) A ConvNet for the 2020s. 2022 *IEEE/CVF Conference on Computer Vision and Pattern Recognition* (*CVPR*), New Orleans, 18-24 June 2022, 11966-11976. https://doi.org/10.1109/cvpr52688.2022.01167

[11] Wang, B., Shang, L., Lioma, C., Jiang, X., Yang, H., Liu, Q. and Simonsen, J.G. (2020) On Position Embeddings in Bert. *International Conference on Learning Representations*. https://doi.org/10.48550/arXiv.2404.10518

[12] Takase, S. and Okazaki, N. (2019) Positional Encoding to Control Output Sequence Length. *Proceedings of the* 2019 *Conference of the North*, Minneapolis, 3-5 June 2019, 3999-4004. https://doi.org/10.18653/v1/n19-1401

[13] Bastidas, A.A. and Tang, H. (2019) Channel Attention Networks. 2019 *IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops* (*CVPRW*), Long Beach, 16-17 June 2019, 881-888. https://doi.org/10.1109/cvprw.2019.00117

[14] Zhu, X., Cheng, D., Zhang, Z., Lin, S. and Dai, J. (2019) An Empirical Study of Spatial Attention Mechanisms in Deep Networks. 2019 *IEEE/CVF International Conference on Computer Vision* (*ICCV*), Seoul, 27 October-2 November 2019, 6687-6696. https://doi.org/10.1109/iccv.2019.00679

[15] Pratt, W.K., Kane, J. and Andrews, H.C. (1969) Hadamard Transform Image Coding. *Proceedings of the IEEE*, **57**, 58-68. https://doi.org/10.1109/proc.1969.6869.