S & M 4121

Light-weight Algorithm for Improving Smoke and Fire Detection Accuracy in Complex Environments

Meiyan Lin,[1*] Chunling Zhang,[2] Wenwu Liu,[3] and Hsien-Wei Tseng[4**]

[1]Concord University College Fujian Normal University, Fuzhou 350117, Fujian, China
[2]Department of Social Sciences and Public Affairs, Quanzhou Medical College, Quanzhou 362011, China
[3]College of Information Engineering, Yango University, Fuzhou 350015, Fujian, China
[4]College of Artificial Intelligence, Yango University, Mawei District, Fujian 350015, China

(Received June 5, 2025; accepted July 4, 2025)

Keywords: YOLOv8n, attention mechanism, loss function, smoke and fire detection, light-weight algorithm

A novel algorithm [high-performance graphical processing unit net version 2-efficient multi-branch and scale feature pyramid network-multipath coordinate attention-you only look once (HGNetV2-EMBSFPN-MPCA-YOLO, HEM-YOLO)] was used in this study to improve smoke and fire detection accuracy in complex environments. By modifying the weights of negative and positive samples, we identified small target classes using improved exponential moving averages with spatial learning loss (EMASlide loss). Multipath coordinate attention (MPCA) was also employed to improve detection accuracy as it efficiently extracted local and global features from images. The backbone network in HGNetV2 enabled more efficient and rapid training of the algorithm. As a result, the enhanced detector head model in EMBSFPN recognized complex patterns to detect and identify smoke and fires in complex environments. The enhanced HEM-YOLO reduced the number of parameters to 1.8 million and floating point operations per second to 6.0 G and increased the accuracy by 4.2% in smoke and fire detection. Its efficiency was further improved, reducing false detections and demonstrating its versatility across multiple applications.

1. Introduction

Fire safety and fire prevention are essential in constructing an effective system for efficient emergency management.(1–3) Therefore, smoke and fire detection technologies are vital for the early identification of fires to minimize potential damage. However, current fire detection systems rely on their capability to monitor temperature, humidity, and smoke.(4–6) However, their effectiveness is influenced by the distance between sensors and the fire's point of origin, often resulting in false alarms for fires occurring at greater distances. To overcome this limitation, video fire detection technology has been used as it enables rapid identification owing to its anti-interference capabilities at a low management cost. Compared with conventionally

[*]Corresponding author: e-mail: lmy_fnu@163.com
[**]Corresponding author: e-mail: hsienwei.tseng@gmail.com
https://doi.org/10.18494/SAM5786

used sensor technologies, video fire detection technology provides intuitive fire-site information, facilitating faster personnel evacuation and more efficient firefighting.(5–7)

In video fire detection technology, static and dynamic features, including flame color, flicker characteristics, smoke texture, shape, and area changes, are extracted. Kong *et al.*(8) developed a logistic regression algorithm to analyze the color difference between the fire and the background and identify the area, color, and other characteristics to calculate the probability that the detected fire was real. Yuan proposed a cumulative motion model for fire detection by combining various images taken on-site, which predicted the direction of smoke movement and enhanced detection robustness.(9)

Previous detection models were developed on the basis of manual feature engineering. The parameters in the models did not appropriately represent complex environments, limiting their accuracy and adaptability in identifying smoke and fires from diverse combustible materials under different lighting conditions and fluctuating airflows. To overcome such limitations, a faster region-based convolutional neural network (FR-CNN), you only look once (YOLO), and a single shot multiBox detector (SSD) have been applied to the models. Li *et al.* combined CNN with a detection transformer (DETR) network for smoke and fire detection and added a normalization-based attention module to more accurately detect small objects.(10) Chaoxia *et al.* adopted a color-based anchor point and global information guidance strategies to improve FR-CNN for smoke and fire detection.(11) On the basis of the improved YOLOv4 model and the convolutional block attention module, Muhammad *et al.*(12) proposed an automatic smoke and fire detection system for the visually impaired.

The effectiveness and precision of the smoke and fire detection system have been enhanced owing to the aforementioned models and networks. However, they cannot monitor fluctuations in the size, shape, and area of smoke and fires in video images accurately. Preset anchor frames are not accurately identified, which hinders the precise capture of fire-related targets. In addition, the complexity of the models leads to low efficiency when they are deployed in embedded systems, low accuracy, and poor real-time performance. Therefore, it is necessary to develop an algorithm with enhanced capability to detect smoke and fires. In this study, an enhanced algorithm was developed for smoke and fire detection by integrating high-performance graphical processing unit net version 2 (HGNetV2), efficient multi-branch and scale feature pyramid network (EMBSFPN), multipath coordinate attention (HEM), and you only look once (YOLO) (HEM-YOLO). The enhanced algorithm is lightweight with an efficient up-sampling module, a global isomeric kernel selection mechanism, and an innovative attention mechanism, ensuring multi-scale feature-weighted fusion and multi-scale efficient convolution. The multipath coordinate attention (MPCA) module is also integrated to train the model and adjust the intersection over union (IoU) loss on the basis of YOLOv8. The developed algorithm reduced the number of parameters by 2.1 G and increased the detection accuracy by 4.2%, which enables broad applicability to various-target identification.

2. YOLOv8 Model

YOLO is a single-stage object detection method that divides input images into multiple grids and assigns bounding boxes and corresponding object classes for each grid.(13–17) YOLO employs the non-maximum suppression (NMS) technique to identify overlapping bounding boxes to enhance detection accuracy. YOLOv8 shows excellent scalability, which is well-suited for the different sizes of hardware and compatible with previous YOLO versions.(6,18–20) Its design enables the intuitive comparison of model performance and rapid deployment and testing in different applications with diverse requirements. YOLOv8 optimizes the features of previous YOLO models owing to its efficient backbone networks, accurate targeting, and enhanced feature fusion. Because of these advantages, YOLOv8 ensures high accuracy and fast reasoning speed for various real-time detection tasks, such as autonomous driving, video surveillance, and industrial inspection.(8,21–23) With its optimized application programming interface (API) capability and enhanced cross-platform compatibility, YOLOv8 simplifies the development and deployment process, making it easy for developers to tackle the challenges of different tasks and hardware. Figure 1 shows the YOLOv8 network structure.

3. HEM-YOLO

3.1 Macrostructure

The architecture of HEM-YOLO comprises a backbone, neck, and head (Fig. 2). The backbone is the initial part of the network responsible for feature extraction. It processes the input image and reduces its spatial resolution while increasing its semantic richness. The backbone's primary role is to generate a set of feature maps at various scales and extract

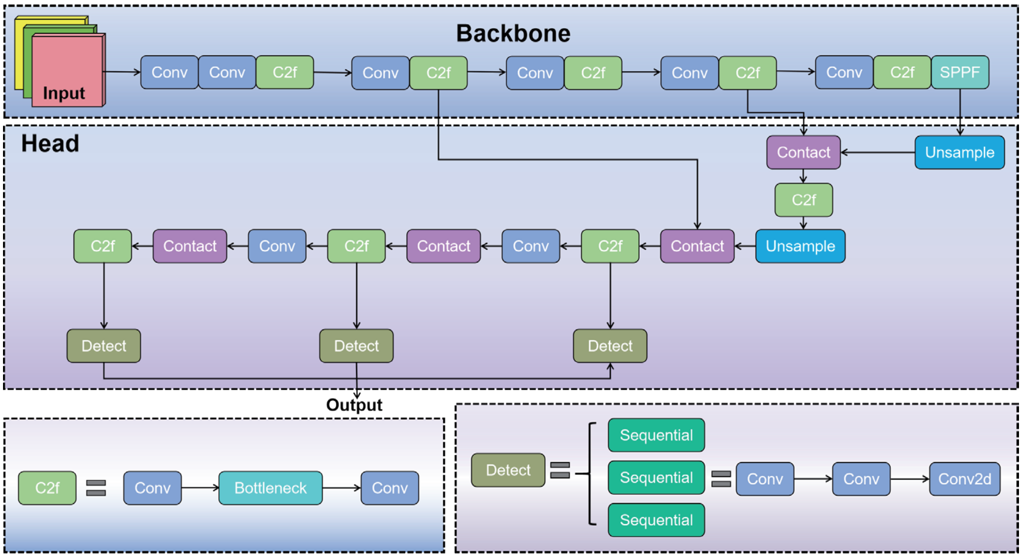


Fig. 1. (Color online) YOLOv8 model's structure.

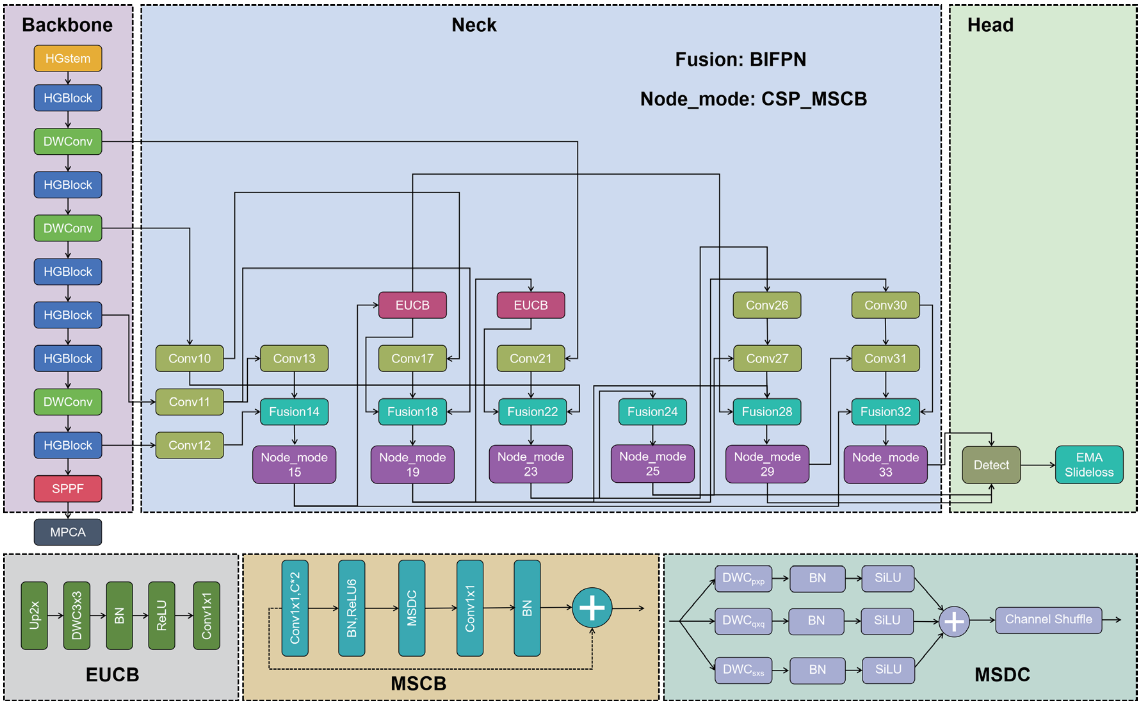


Fig. 2. (Color online) HEM-YOLO model structure in this study.

hierarchical visual representations from the input image. These feature maps are input to the subsequent neck. The input image is processed by the backbone, which consists of five stages: StemBlock[0], HGStage[1], HGStage[2], HGStage[3], and HGStage[4].

The neck is responsible for feature fusion and aggregation. It takes the multi-scale feature maps from the backbone and combines them in a way that enhances information flow and enriches the features for detection. The neck builds a robust and semantically rich feature pyramid that integrates information from different scales of the backbone's output. This allows the head to detect objects of various sizes effectively. In the neck structure with EMBSFPN, the information obtained by the backbone is transmitted to the convolutional layer via HGBlock in the first bottom-up path. In the second path, the fusion module transmits the information to Node_mode through bidirectional feature fusion and weighted feature aggregation. In the Node_mode module, multi-scale convolution is performed to obtain features on different scales, thereby enhancing the representation capability of multi-scale information. On the other hand, several features are reused in different multi-scale convolution block (MSCB) layers by leveraging the structure characteristics of the cross-stage partial network (CSPNet), which preserves the feature flow and effectively reduces calculation. The efficient up-convolution block (EUCB) module enhances the feature representation capability by employing channel extension, convolution update, and feature fusion. The mass storage device class (MSDC) module is used to extract features of multiple scales by using a multi-scale convolution operation in the Node_mode and combining convolution kernels of different sizes. Finally, the target bounding boxes and their corresponding classes are predicted in the detection head on the basis of the feature map on each scale and calculated losses.

The head is the final part of the network responsible for prediction. The head processes and fuses features obtained from the neck and translates them into meaningful object detections, including bounding box coordinates, confidence scores, and class probabilities for each detected object. The exponential moving average (EMA) and slidelessness are used to ensure the robustness and applicability of these predictions. The head generates the final detection outputs. The head of the YOLOv8 algorithm is used for dimensional mapping and processing features in the convolutional procedure. In the HEM-YOLO model structure, HGNetV2 replaces the YOLOv8 backbone, and the multi-branch auxiliary fusion (MAF)-YOLO structure is used to design the neck. MPCA is used to enhance the learning capability of the model.

3.2 Backbone structure

HGNetV2 integrated into the backbone network of YOLOv8 uses graph neural networks (GNNs) and time series analysis to enhance target detection performance. Graph convolution is a core operation in GNNs, aggregating information obtained from adjacent nodes and updating the features of each node. The formula is

$$H^{(l+1)} = \sigma\left(\hat{A}H^{(l)}W^{(l)}\right), \tag{1}$$

where $H^{(l)}$ is the characteristic matrix of the layer l node, $\hat{A}$ is the normalized adjacency matrix (including self-ring), $W^{(l)}$ is the weight matrix of layer l, and σ is the activation function.

HGNetV2 integrates EMBSFPN, which utilizes convolution cores on different scales in the feature layer to efficiently acquire multi-scale perceptual field information, which is obtained from the trident network for the accurate detection of objects of various sizes. The network enhances feature extraction in a series of convolutional layers, from HGStem to HGBlock, and then the spatial pyramid pooling-fast (SPPF) module. Moreover, the MPCA module further improves attention and maintains temporal consistency. In the feature pyramid stage, multi-scale feature weighted fusion is carried out in the bi-directional feature pyramid network (BIFPN), which replaces the concatenation operation with the addition operation, thereby reducing the number of parameters and improving computational efficiency. EUCB is applied to nodes in the convolutional layer for efficient upsampling, which is critical for feature layers conducting fusion in the Node_mode module. The network outputs multilevel detection headers that enable efficient object detection through the convolutional to fusion layers. Integrating HGNetV2 with the YOLOv8 framework and EMBSFPN enables a lightweight, efficient, and effective multi-scale feature fusion network, which significantly improves the accuracy of target detection (Fig. 3).

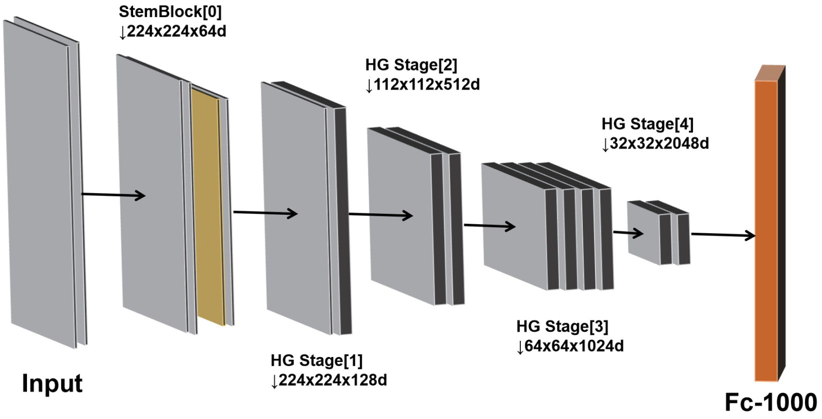


Fig. 3. (Color online) Structure of HGNetV2.

3.3 Head structure

On the basis of the global isomeric kernel selection mechanism of MAF-YOLO, EUCB and the multi-scale feature weighted fusion in BIFPN are used for reference.(24–26) If the concatenation operation is replaced with the addition operation to reduce computational complexity and the number of parameters, self-applicable selection weighted fusion is carried out depending on the importance of features on different scales to improve the accuracy and efficiency of detection. The information flow of BIFPN is exchanged during up- and down-sampling using Eq. (2).

$$F_{fusion} = \sum_{i=1}^{n} W_i \cdot F_i \,, \tag{2}$$

where W_i and F_i represent the weighting coefficient and the feature map of layer i, respectively.

In the following recursion operations, F_{high} and F_{low} represent the high-level and low-level feature maps, respectively.

$$F_{high} = Upsample\left(F_{low}\right) + F_{high} \tag{3}$$

$$F_{low} = Downsample\left(F_{high}\right) + F_{low} \tag{4}$$

The head structure utilizes a series of convolution layers to extract and optimize feature maps that play a key role in downsampling and capturing contextual information (Fig. 4). The C2f layer is integrated to ensure smooth transitions between different feature scales and seamlessly transmits information to the various stages of the network. In the head structure, the Node_mode module is introduced to transform specific features, providing dynamic adaptability to the feature map and making its representation flexible and adaptable. In fusion, features from

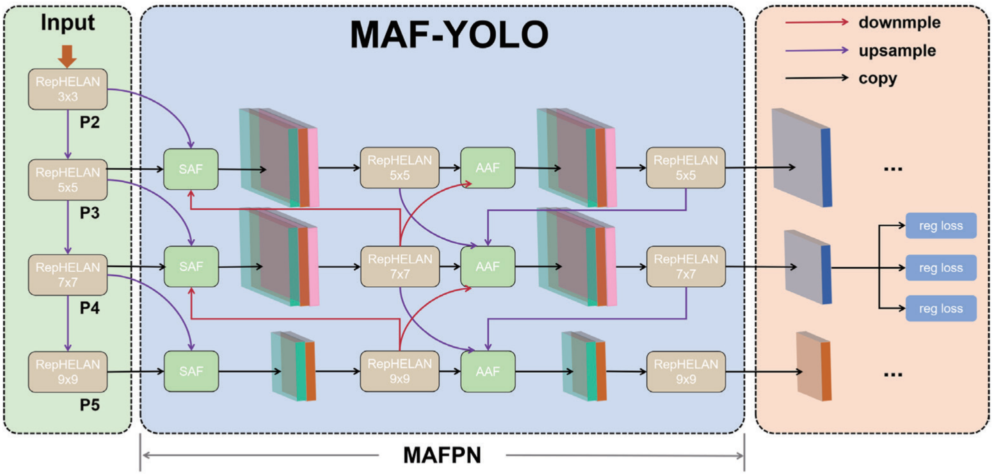


Fig. 4. (Color online) Structure of the MAF-YOLO

different sources are connected and combined, enriching feature information and diversity. Subsequently, the EUCB module optimizes the feature map by recalibrating channel features, emphasizing informative channels while suppressing less relevant features.(2,27–30) Combined with an efficient upsampling technique and accurate feature calibration, this mechanism greatly enhances feature differentiation. On the other hand, the EUCB module improves the spatial resolution of feature maps, ensuring that the most critical features are extracted, thus improving the object detection and classification capabilities of the model. CSPNet improves computational efficiency by splitting the feature graph F into two, F_1 and F_2, and processing them separately.

$$F_1, F_2 = Split\left(F\right) \tag{5}$$

Through different convolution operations, the following are obtained.

$$F_1' = Conv\left(F_1\right) \tag{6}$$

$$F_2' = Conv\left(F_2\right) \tag{7}$$

The processed F_1' and F_2' are merged as

$$F_{csp} = Concat\left(F_1', F_2'\right). \tag{8}$$

The MSCB layer uses convolution kernels of different sizes to handle different scales of feature maps. For the multiple convolution kernels of sizes 3 × 3, 5 × 5, and 7 × 7 (K_1, K_2, and K_3, respectively), the multi-scale convolution operation is conducted as follows.

$$F_{MSC} = \sum_{k=1}^{3} Conv\left(F, K_k\right), \tag{9}$$

where K_k is the size of the convolution kernel, and $Conv(F,K_k)$ represents a convolution operation with a convolution kernel of size K_k on the input feature map F.

The head structure also includes a detection layer that predicts target bounding boxes and class probabilities, which is essential for converting the optimized feature map into an actionable output for accurate target detection. In the layer, the category prediction is performed using Eqs. (10) and (11).

$$C = Soft_{max}\left(W_{box} \cdot F_{fusion} + b_{box}\right) \tag{10}$$

$$B = sig_{moid}\left(W_{box} \cdot F_{fusion} + b_{box}\right) \tag{11}$$

3.4 Loss functions

Unbalanced training samples still exist in bounding box regression through the process. Traditional techniques are used to address this imbalance, but they require resampling and reweighting samples during training despite their limited effectiveness.(28–30) Therefore, a focal loss is used to solve this issue by highlighting readily identifiable negative samples in the overall loss and modifying the slope appropriately.(23) By changing the weights of positive and negative samples, the model enhances differentiation among samples in uncommon target categories. In other words, negative samples that are reasonably easy to classify are less weighted, whereas positive samples that are challenging to classify are prioritized. In target size analysis, observable targets are considered negative samples, whereas microscopic targets are considered positive samples owing to their difficulty in localization.(7,31,32) Negative samples are detected effectively as they are optimized in bounding box regression.(33,34) In contrast, the model needs to process positive samples to improve its overall performance, which increases computational complexity. To solve this problem, improved exponential moving averages with spatial learning loss (EMASlide loss) are introduced as they enable the creation of a smoother loss curve with more stable convergence, reducing the influence of outliers on training and improving the model's generalization capability.

$$iou\_mean(t) = \alpha\left(t\right) \cdot iou\_mean(t-1) + \left(1-\alpha\left(t\right)\right) \cdot auto\_iou(t), \tag{12}$$

$$\alpha\left(t\right) = decay(t) = decay \cdot \left(1-\exp(\tau - t)\right), \tag{13}$$

where $\alpha(t)$ is the attenuation factor, which gradually decreases with increasing number of training steps t, and decay is a fixed attenuation coefficient, which is set to 0.999 to ensure a long smoothing period without immediately reducing the influence of historical values. *auto_iou*(*t*) is the average IoU value of the current batch sample, and *iou_mean*(*t*) is the IoU value after the sliding average.

3.5. Attention mechanism

In the HEM-YOLO structure, the MPCA module is designed to enhance feature processing. The module starts from the input feature graph.

Firstly, $X_h \in \mathrm{R}^{C\times H\times 1}$, $X_w \in \mathrm{R}^{C\times 1\times W}$, and global average pooling (GAP) $X \in \mathrm{R}^{C\times H\times W}$ are obtained by the average pool in the *X* and *Y* directions. Then, the global features are extracted with different spatial information to obtain X_{ch} = GAP(X) $\in \mathrm{R}^{C\times 1\times 1}$. The features are integrated and rearranged to form a richer feature representation. Feature fusion is conducted in the concatenation and permuting operations to concatenate X_h and X_w in the channel dimension to obtain $X_{hw} \in \mathrm{R}^{C\times (H+W)\times 1}$. The combined information is extracted through a convolution operation to obtain the fused feature graph $X_{hw}' \in \mathrm{R}^{C\times (H+W)\times 1}$ and then refined in the convolution layer. In this process, multiple layers of convolutional operations capture more important features while maintaining computational efficiency. Furthermore, the features extracted in the convolution layer are nonlinearly transformed by the Sigmoid activation function. The features are then divided and averaged to generate the attention weight $A_{hw} = \sigma(\mathrm{Conv}1\times 1(X_{hw}')) \in \mathrm{R}^{C\times (H+W)\times 1}$, which is divided into $A_h \in \mathrm{R}^{C\times H\times 1}$ and $A_w \in \mathrm{R}^{C\times 1\times W}$. The weight is used to recalibrate the input features in subsequent multiplication operations, providing emphasized *X* and *Y* direction feature weights *(X* Weight and *Y* Weight in Fig. 5). For the final feature output, the weighted features in the *X* and *Y* directions are multiplied by the original features. A_h and A_w are applied to X_h and X_w. On the other hand, the average value of A_{hw} is weighted to the global feature X_{ch}. The most valuable information is extracted by obtaining $X_h' = X_h{\cdot}A_h$, $X_w' = X_w{\cdot}A_w$, and $X_{ch}' = X_{ch}{\cdot}\mathrm{mean}(A_{hw})$. Through further multiplication, these enhanced features are integrated to generate the output of the MPCA module, enhancing the reasoning power of the entire model as follows.

$$MPCA(X) = X\cdot(X_h{}')\cdot\sigma(X_w{}')\cdot\sigma(X_{ch}{}') \tag{14}$$

In the modular design (Fig. 5), the MPCA module improves the capability to distinguish features and ensures efficient and effective computation. This module plays a key role in feature selection and enhancement and contributes to the improvement in the overall performance of the model.

3.6 Dataset, equipment, and evaluation

In the experiment, a well-constructed dataset composed of a large number of high-dimensional images is required so that the model can extract rich and diverse features for accurate smoke and fire detection. Therefore, we collected images online from the flame and smoke detection dataset (FASDD), ‘FireAndSmokeDataset1’, and ‘Fire Smoke DetectionDataset2’, which contain images of fire and smoke.[35,36] The dataset constructed in this study was filtered

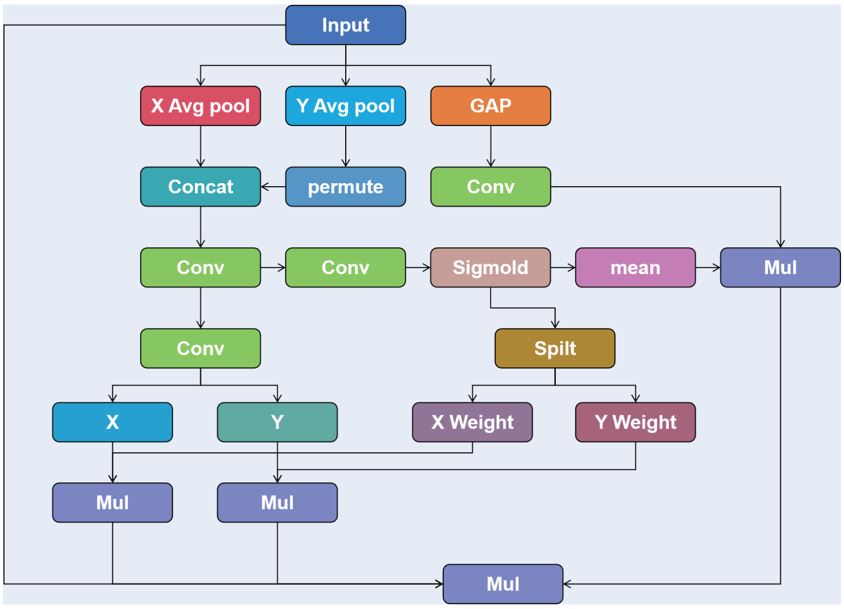


Fig. 5. (Color online) MPCA structure.

and annotated using LabelImg software. 1200 images were split randomly into the training, test, and validation sets at a ratio of 7:2:1. An online data augmentation library, Albumentations, was used for median blur, blurring, and grayscale transformation. Contrast-limited adaptive histogram equalization was applied to enhance the robustness, feature extraction capability, contrast enhancement, denoising capability, and data diversity of the model.

The model was created in Python in the PyTorch deep learning framework. PyTorch is an open-source machine learning framework developed by Meta AI, widely recognized for its flexibility and strong graphic processing unit (GPU) acceleration. It is widely used in deep learning research and development.(37) Compute unified device architecture (CUDA) 12.6.65 was used to speed up the training process. CUDA is a parallel computing platform and API developed by NVIDIA. It enables the use of NVIDIA GPUs, a paradigm known as general-purpose computing on graphics processing units (GPGPU). CUDA is the backbone of GPU acceleration for deep learning frameworks such as PyTorch.(38) The hardware used in the experiment included a 12th-generation Intel® Core™ i3-12100F CPU and an NVIDIA RTX 4060 Ti GPU with 8 GB of video memory. In the training, the input image size was set as 640 × 640 pixels, and the optimizer with weight decay in stochastic gradient descent was employed. The model was trained for 200 epochs with an early stop strategy to prevent overfitting. The batch size was 32, and the initial learning rate was 0.01.

Mean average precision (*mAP*) was calculated to evaluate model performance. The complexity of the model was measured in giga floating point operations per second (GFLOPs), whereas the model size was evaluated using the parameters. Generally, the smaller the number of parameters and GFLOPs, the lower the computational load on the model.

4. Results and Discussion

4.1 Effectiveness of enhanced attention mechanism

To evaluate the effectiveness of the MPCA module, YOLOv8n was used as the backbone network. Other attention mechanisms, such as SpatialGroupEnhance, the lattice-structured kernel block (LSKBlock), and the simple attention module (SimAM), were integrated into the SPPF module. These attention mechanisms were used to train the model and verify the purposes at the backbone network. Table 1 presents the target parameters for effectiveness evaluation.

The performances of the models with different attention mechanisms were evaluated by comparing detection performance (Table 2).

Table 1
Target parameters of effectiveness evaluation of attention mechanism.

Model	Number of parameters (million)	FLOPs (GigaFLOPs)	Model (Megabyte)	*mAP* (%)	Precision (%)	Recall (%)
YOLOv8n	3.01	8.1	6.2	51.1	51.9	50.3
YOLOv8n-SimAM	3.01	8.1	6.2	54.9	57.9	50.5
YOLOv8n-spatial group-wise enhance (SGE)	3.01	8.1	6.2	53	57.8	51.3
YOLOv8n-transformer-based prediction attention (TPA)	3.01	8.1	6.2	53.3	56.3	52
YOLOv8n-LSKB	3.26	8.1	6.5	52.3	56.8	46.9
YOLOv8n-AFGC	3.07	8.1	6.3	52.4	52.2	50.4
YOLOv8n-extended finite-state machine (EFSM)	3.07	8.1	6.3	55.3	57.2	50
YOLOv8n-MPCA	3.33	8.1	6.6	56.4	61.3	53.8

Table 2
Performance comparison of different models.

Model	Number of parameters (million)	FLOPs (GigaFLOPs)	Model (Megabyte)	*mAP* (%)	Precision (%)
YOLOv8n	3.01	8.1	51.1	51.9	50.3
YOLOv8n-HGNetV2	2.35	6.9	45.8	42.6	48.4
YOLOv8n-EMBSFPN	2.12	7.1	55.1	54.9	53.8
YOLOv8n-MPCA	3.33	8.1	56.4	61.3	53.8
YOLOv8n-EMASL	3.01	8.1	52.9	52.8	50.5
YOLOv8n-EMBSFPN-MPCA (HEM-YOLO)	1.80	6.0	55.3	59.6	48.7

In training, the YOLOv8n models with different attention mechanisms showed various *mAP*s. The HEM-YOLO model used the EMASlide loss function instead of the IoU loss function with an *mAP* of 59.6% with a smaller number of parameters and FLOPs. The EMASlide loss enhanced accuracy. The YOLOv8n-EMBSFPN model optimized the head part and reduced the number of parameters and FLOPs by 0.89 M and 1.0, respectively, resulting in a 4% increase in *mAP* and a 3% increase in accuracy compared with YOLOv8n. The YOLOv8n-EMBSFPN model reduced the complexity of the model while improving its detection accuracy. The backbone structure of the YOLOv8n-HGNetV2 model was modified to reduce the number of parameters by 0.66 M and FLOPs by 1.2 G, indicating that the model structure was lighter and more efficient. The YOLOv8n-MPCA model with an improved attention mechanism increased the number of parameters by 0.32 M with constant FLOPs. This resulted in a 5.3% improvement in *mAP*, a 9.4% improvement in accuracy, and a 3.5% improvement in recall. Compared with the original YOLOv8n model, the number of parameters of the HEM-YOLO model was reduced by 1.21M, *mAP* was increased by 4.2%, and the accuracy was increased by 7.7%. The HEM-YOLO model improved precision and reduced the weight of the model significantly.

Table 2 shows that the HEM-YOLO model showed an *mAP* of 59.6% and a precision of 48.7%, which indicated that the model's capability was enhanced to correctly identify and locate fires, reducing missed fires and false alarms. HEM-YOLO achieved a higher *mAP* than the other models, indicating superior fire detection. Real-time responsiveness is directly related to a lower number of parameters, FLOPs, and model size. The number of parameters, FLOPs, and size of the HEM-YOLO model were 1.80 million, 6.0 GigaFLOPs, and 55.3 megabytes, respectively, which showed that the model has a significantly reduced size and computational complexity. The HEM-YOLO model processed images much faster than other models, making it highly appropriate for applications where immediate fire detection is critical, such as surveillance systems, drones, or edge devices. The HEM-YOLO model was significantly lighter and more efficient than the other models but processed the image in milliseconds to send an almost instantaneous alert for quicker intervention. The rapid response, efficient deployment, enabling extended operational time, and accurate detection capability make the HEM-YOLO model well suited for small devices, such as drones, as it can extend the patrol duration and ensure continuous monitoring before significant battery draining occurs.

4.2 Ablation experiment

To examine the effect of improved models on the detection of smoke and fire, ablation experiments were conducted with the same parameters. The YOLOv8n and YOLOv8n-EMASL models had enhanced loss functions, the YOLOv8n-EMBSFPN model had improved head structures, the YOLOv8n-HGNetV2 model had an enhanced backbone network, and the YOLOv8n-MPCA model had improved attention mechanisms. The results are illustrated in Table 3, in which '√' indicates the improvement.

The YOLOv8n algorithms with enhanced attention mechanisms outperformed the original YOLOv8 model. The *mAP* was improved by 4.2% on average. Although FLOPs decreased by 2.1 G, the number of parameters decreased by 1.21 M. The ablation experiment results verified the

Table 3
Comparison of results of ablation experiments using different models.

YOLOv8n	YOLOv8n-HGNetV2	YOLOv8n-EMBSFPN	YOLOv8n-MPCA	YOLOv8n-EMASL	Number of parameters (million)	FLOPs (GigaFLOPs)	*mAP* (%)
√					3.01	8.1	51.1
√	√				2.35	6.9	45.8
√	√	√			1.47	5.9	43.6
√	√	√	√		1.8	6.0	51.7
√	√	√	√	√	1.8	6.0	55.3

effectiveness of the models with enhanced attention mechanisms compared with the traditional YOLOv8n model, emphasizing the advantages and applications of the modified models.

The YOLOv3, YOLOv5, YOLOv6, and YOLOv8 models were trained using the same dataset with their hyperparameters and training parameters. Their results were compared with those of the HEM-YOLO model developed in this study. The results shown in Table 4 indicate that the HEM-YOLO model improved *mAP* by 7.7%. In addition, while optimizing FLOPs, the number of parameters, precision, and recall were also enhanced.

The training results of the YOLOv8n and HEM-YOLO models are shown in Fig. 6. In training, the accuracy of the HEM-YOLO model was higher than that of the YOLOv8n model. The HEM-YOLO model completed training after 90 epochs using the early stop method, whereas the YOLOv8n model required more than 140 epochs for training and learning. The precision of the HEM-YOLO model was higher than that of the YOLOv8n model.

Figure 7 verifies the better performance of the HEM-YOLO model than of the YOLOv8n model. The YOLOv8n model failed to detect incipient fires, whereas the HEM-YOLO model detected them under the same conditions with a higher detection precision. The HEM-YOLO model conducted feature fusion and context awareness better and improved overall detection by enabling precise location and identification. The HEM-YOLO model assigned higher confidence scores to its detections than the YOLOv8n model for the same fire instances. For example, in the third row, the second image with a truck with fire, YOLOv8n detected “fire 0.3”, whereas HEM-YOLO detected “fire 0.9”. This higher confidence indicated that the HEM-YOLO model was more certain about its detections, reducing ambiguity. For the small fire on the truck (third row, second image), the HEM-YOLO model’s bounding box accurately encompasses the fire, whereas the YOLOv8n model showed vague or lower confidence. Although not explicitly showing false negatives, the HEM-YOLO model showed better detections and was less likely to miss fire instances at a lower false negative rate and higher recall.

Figure 8 presents heat maps, which visualize where the model “looks” or focuses its attention to identify a fire. Areas with higher intensity (red/yellow) indicate where the model’s attention is concentrated. In the first image pair (hand with fire source), the YOLOv8n model’s heatmap shows a diffused attention, spread around the hand and the fire source. In contrast, the HEM-YOLO model’s heatmap shows more concentrated and localized features on the fire source (the bright flame). This indicates that the HEM-YOLO model is more adept at isolating the critical “fire” features from the background or surrounding elements. The sharper, more focused heat

Table 4
Comparison of results of different YOLO models.

Model	Number of parameters (million)	FLOPs (GigaFLOPs)	Model (Megabyte)	*mAP* (%)	Precision (%)
YOLOv3	12.13	18.9	52.5	48.1	49.2
YOLOv5	2.50	7.1	50.6	48.5	46
YOLOv6	4.23	11.8	49.7	47.3	49.8
YOLOv8n	3.01	8.1	51.1	51.9	50.3
HEM-YOLO	1.80	6.0	55.3	59.6	48.7

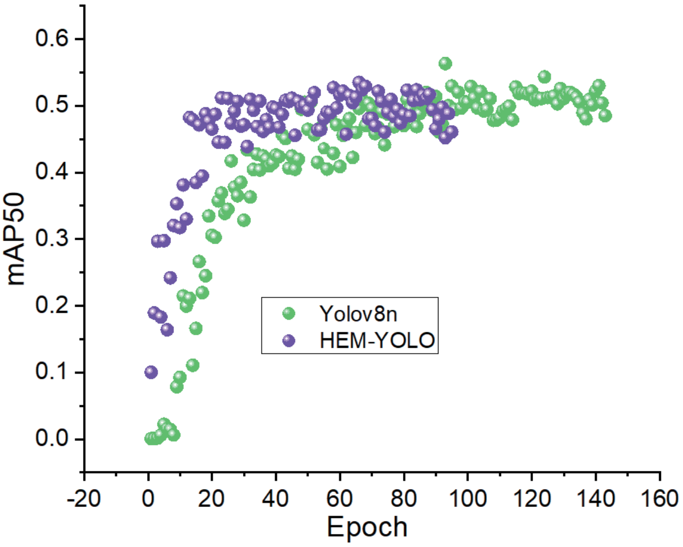
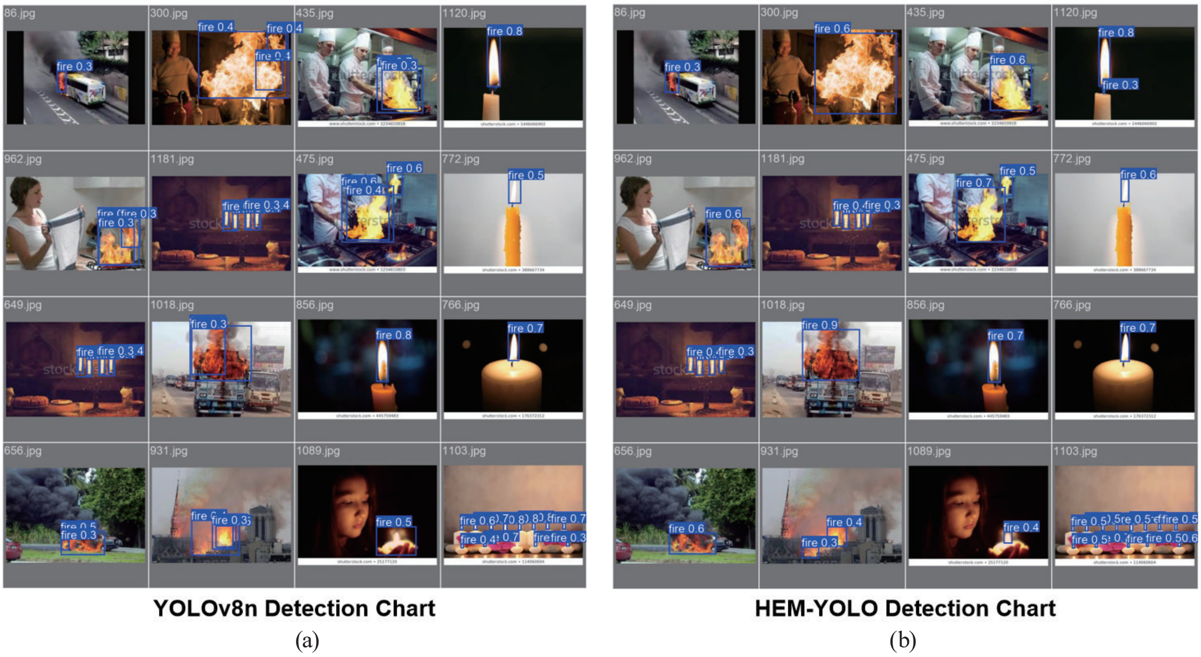


Fig. 6. (Color online) Comparison of *mAP* between YOLOv8n and HEM-YOLO models.



YOLOv8n Detection Chart
(a)

HEM-YOLO Detection Chart
(b)

Fig. 7. (Color online) Comparison of detection results between (a) YOLOv8n and (b) HEM-YOLO models.

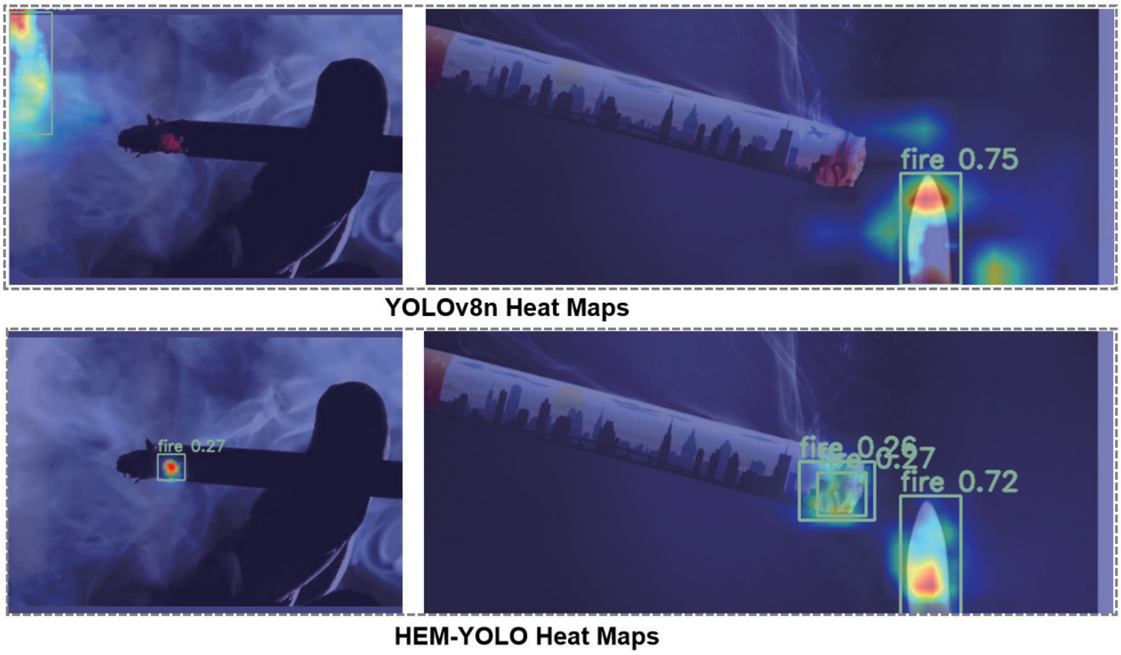


Fig. 8. (Color online) Comparison of heat maps of YOLOv8n and HEM-YOLO models.

maps of the HEM-YOLO model suggest its superior capability to distinguish fire features from other visually similar elements, which is crucial for reducing false positives and improving precision.

By focusing more precisely on the fire, the HEM-YOLO model was more robust to cluttered backgrounds or challenging lighting conditions, as it can effectively filter out irrelevant information. Figures 7 and 8 show the quantitative improvements in the HEM-YOLO model's capability to provide higher confidence, more accurate bounding boxes, and more vivid heat maps. This shows the superiority of the HEM-YOLO model in fire detection over the YOLOv8n model. The HEM-YOLO model provides a better visualization of incipient fires, offering a refined and specific distribution that enhances the detection and representation of small-scale fires. The model also consistently demonstrates superior detection accuracy, better localization, and enhanced focus on relevant features for fire detection compared with the baseline YOLOv8n model. This leads to more reliable and precise fire detection.

5. Conclusions

In this study, an enhanced smoke and fire detection algorithm, the HEM-YOLO model, was developed to address the limitations of traditional sensor- and video-based detection technologies, particularly in complex environments with poor image resolution and difficult feature extraction. On the basis of YOLOv8, the HEM-YOLO model incorporates multiple improvements, such as HGNetV2 as a lightweight yet efficient backbone network, the EMNSFPN detector head for complex pattern recognition, and MPCA for effectively capturing local and global features. Additionally, the EMASlide loss is employed to modify the weights of negative and positive samples to enhance the recognition of small target classes. These enhancements collectively improve the model's adaptability, detection accuracy, precision, and recall under various environmental conditions, including diverse lighting and airflow. Ablation experiment results confirmed the individual and combined contributions of each component to overall model performance. Compared with the YOLOv8n models with different attention mechanisms, the HEM-YOLO model demonstrated a 4.2% increase in *mAP*, a 7.7% boost in accuracy, and a reduction of 1.21M parameters, validating its efficiency and lightweight design. Moreover, it meets real-time detection requirements while significantly reducing computing and storage demands, facilitating deployment on resource-constrained platforms. These results highlight the HEM-YOLO model's potential for applications in emergency management, industrial safety monitoring, and intelligent surveillance. In future research, expanded datasets need to be used for the model's optimization in embedded and edge computing environments to improve its scalability and adaptability.

Acknowledgments

This research was supported by the Project of Fujian Provincial Department of Education (No.JAT220476) and the Scientific Research Funds of Quanzhou Medical College (No. XJY2414).

References

1 Y. Li, W. Zhang, Y. Liu, R. Jing, and C. Liu: Eng. Appl. Artif. Intell. **116** (2022) 105492. https://doi.org/10.1016/j.engappai.2022.105492
2 Z. Yang, Q. Guan, K. Zhao, J. Yang, X. Xu, H. Long, and Y. Tang: Proc. 2024 Pattern Recognition and Computer Vision: 7th Chinese Conf. (2024) 492. https://doi.org/10.1007/978-981-97-8858-3_34
3 M. M. Rahman, M. Munir, and R. E. Marculescu: Proc. 2024 IEEE/CVF Conf. Computer Vision and Pattern Recognition (IEEE, 2024) 11769. https://doi.org/10.48550/arXiv.2405.06880
4 H. Lou, X. Duan, J. Guo, H. Liu, J. Gu, L. Bi, and H. Chen: Electronics **12** (2023) 2323. https://doi.org/10.3390/electronics12102323
5 Y. Zhao, W. Lv, S. Xu, J. Wei, G. Wang, Q. Dang, Y. Liu, and J. Chen: Proc. 2024 IEEE/CVF Conf. Computer Vision and Pattern Recognition (IEEE, 2024) 16965. https://doi.org/10.48550/arXiv.2304.08069
6 Q. Hou, D. Zhou, and J. Feng: Proc. 2021 IEEE/CVF Conf. Computer Vision and Pattern Recognition (IEEE, 2021) 13713. https://doi.org/10.48550/arXiv.2103.02907
7 T. Y. Ross and G. K. H. P. Dollár: Proc. 2017 IEEE Conf. Computer Vision and Pattern Recognition (IEEE, 2017) 2980. https://doi.org/10.1109/TPAMI.2018.2858826
8 S. G. Kong, D. Jin, S. Li, and H. Kim: Fire Saf. J. **79** (2016) 37. https://doi.org/10.1016/j.firesaf.2015.11.015
9 F. Yuan: Pattern Recognit. Lett. **29** (2008) 925. https://doi.org/10.1016/j.patrec.2008.01.013
10 S. Li, Q. Yan, and P. Liu: IEEE Trans. Image Process. **29** (2020) 8467. https://doi.org/10.1109/TIP.2020.3016431
11 C. Chaoxia, W. Shang, and F. Zhang: IEEE Access **8** (2020) 58923. https://doi.org/10.1109/ACCESS.2020.2982994
12 K. Muhammad, S. Khan, M. Elhoseny, S. H. Ahmed, and S. W. Baik: IEEE Trans. Ind. Inform. **15** (2019) 3113 https://doi.org/10.1109/TII.2019.2897594
13 M. Tan, R. Pang, and Q. V. Le: Proc. 2020 IEEE/CVF Conf. Computer Vision and Pattern Recognition (IEEE, 2020). https://doi.org/10.48550/arXiv.1911.09070
14 D. Shi, Y. Zhong, Q. Cao, L. Ma, J. Li, and D. Tao: Proc. 2023 IEEE/CVF Conf. Computer Vision and Pattern Recognition (IEEE, 2023). https://doi.org/10.48550/arXiv.2303.07347
15 C. Y. Wang, H. Y. M. Liao, Y. H. Wu, P. Y. Chen, J. W. Hsieh, and I. H. Yeh: Proc. 2020 IEEE/CVF Conf.

Computer Vision and Pattern Recognition (IEEE, 2020) 390. https://doi.org/10.48550/arXiv.1911.11929
16 J. Zhan, Y. Hu, W. Cai, G. Zhou, and L. Li: Symmetry **13** (2021) 2260. https://doi.org/10.3390/sym13122260
17 L. Zhao, L. Zhi, C. Zhao, and W. Zheng: Sustainability **14** (2022) 4930. https://doi.org/10.3390/su14094930
18 C. Ho: Meas. Sci. Technol. **20** (2009) 045502. https://doi.org/10.1088/0957-0233/20/4/045502
19 Y. Yang, S. Hu, and Y. Ke: Int. J. Intell. Comput. **16** (2023) 502. https://doi.org/10.1108/IJICC-11-2022-0291
20 M. Ji and P. Zhao: Signal Image Video Process. **17** (2023) 1733. https://doi.org/10.1007/s11760-022-02384-z
21 J. Zhan, Y. Hu, and G. Zhou: Comput. Electron. Agric. **196** (2022) 106874. https://doi.org/10.1016/j.compag.2022.106874
22 B. Ding, Y. Zhang, and S. Ma: Drones **8** (2024) 479. https://doi.org/10.3390/drones8090479
23 S. Kim and S. Park: Appl. Soft Comput. **165** (2024) 112037. https://doi.org/10.1016/j.asoc.2024.112037
24 M. O. Almasawa, L. A. Elrefaei, and K. Moria: IEEE Access **7** (2019) 175228 https://doi.org/10.1109/ACCESS.2019.2957336
25 Z. Ming, M. Zhu, X. Wang, J. Zhu, J. Cheng, C. Gao, Y. Yang, and X. Wei: Image Vis. Comput. **119** (2022) 104394. https://doi.org/10.1016/j.imavis.2022.104394
26 Z. Zhang, H. Zhang, and S. Liu: Proc. 2021 IEEE/CVF Conf. Computer Vision and Pattern Recognition (IEEE, 2021) 12131. https://doi.org/10.1109/CVPR46437.2021.01196
27 H. Touvron, M. Cord, M. Douze, F. Massa, A. Sablayrolles, and H. Jégou: arXiv:2012.12877 (2021). https://doi.org/10.48550/arXiv.2012.12877
28 Y. An, J. Wu, Y. Cui, and H. Hu: IEEE Trans. Veh. Technol. **72** (2023) 9909. https://doi.org/10.1109/TVT.2023.3259999
29 D. N. Truong Cong, C. Achard, and L. Khoudour: Proc. 2010 Int. Conf. Image Processing Theory, Tools and Applications (IPTA, 2010) 60. https://doi.org/10.1109/IPTA.2010.5586809.
30 Z. Zhang, C. Lan, W. Zeng, X. Jin, and Z. Chen: arXiv:1904.02998 (2020). https://doi.org/10.48550/arXiv.1904.02998
31 H. Luo, Y. Gu, X. Liao, S. Lai, and W. Jiang: Proc. 2019 IEEE/CVF Conf. Computer Vision and Pattern Recognition Workshops (IEEE, 2019) 1487. https://doi.org/10.1109/CVPRW.2019.00190
32 X. Jin, C. Lan, W. Zeng, and G. Wei: Proc. 2020 AAAI Conf. Artificial Intelligence (AAAI, 2020) 1117. https://doi.org/10.1609/aaai.v34i07.6775
33 H. Yao, S. Zhang, R. Hong, Y. Zhang, C. Xu, and Q. Tian: IEEE Trans. Image Process. **28** (2019) 2860. https://doi.org/10.1109/TIP.2019.2891888
34 L. Zheng, L. Shen, L. Tian, S. Wang, J. Wang, and Q. Tian: Proc. 2015 IEEE Int. Conf. Computer Vision (ICCV, 2015) 1115. https://doi.org/10.1109/ICCV.2015.133
35 GitHub: https://github.com/openrsgis/FASDD (accessed June 2025).
36 Roboflow: https://universe.roboflow.com/browse/fire (accessed June 2025).
37 NVIDIA: https://www.nvidia.com/en-us/glossary/pytorch/ (accessed June 2025).
38 NVIDIA: https://developer.nvidia.com/cuda-zone (accessed June 2025).