Computers, Materials & Continua

Tech Science Press

DOI: 10.32604/cmc.2024.057954

ARTICLE

Enhancing Fire Detection Performance Based on Fine-Tuned YOLOv10

Trong Thua Huynh*, Hoang Thanh Nguyen and Du Thang Phu

Faculty of Information Technology 2, Posts and Telecommunications Institute of Technology (PTIT), Ho Chi Minh City, 700000, Vietnam

*Corresponding Author: Trong Thua Huynh. Email: thuaht@ptit.edu.vn

Received: 31 August 2024 Accepted: 17 October 2024 Published: 18 November 2024

ABSTRACT

In recent years, early detection and warning of fires have posed a significant challenge to environmental protection and human safety. Deep learning models such as Faster R-CNN (Faster Region based Convolutional Neural Network), YOLO (You Only Look Once), and their variants have demonstrated superiority in quickly detecting objects from images and videos, creating new opportunities to enhance automatic and efficient fire detection. The YOLO model, especially newer versions like YOLOv10, stands out for its fast processing capability, making it suitable for low-latency applications. However, when applied to real-world datasets, the accuracy of fire prediction is still not high. This study improves the accuracy of YOLOv10 for real-time applications through model fine-tuning techniques and data augmentation. The core work of the research involves creating a diverse fire image dataset specifically suited for fire detection applications in buildings and factories, freezing the initial layers of the model to retain general features learned from the dataset by applying the Squeeze and Excitation attention mechanism and employing the Stochastic Gradient Descent (SGD) with a momentum optimization algorithm to enhance accuracy while ensuring real-time fire detection. Experimental results demonstrate the effectiveness of the proposed fire prediction approach, where the YOLOv10 small model exhibits the best balance compared to other YOLO family models such as nano, medium, and balanced. Additionally, the study provides an experimental evaluation to highlight the effectiveness of model fine-tuning compared to the YOLOv10 baseline, YOLOv8 and Faster R-CNN based on two criteria: accuracy and prediction time.

KEYWORDS

Fire detection; accuracy; prediction time; fine-tuning; real-time; YOLOv10; Faster R-CNN

1 Introduction

The situation of fires and explosions in buildings, factories, and warehouses is becoming increasingly complex and severe, causing significant damage to people and property. Therefore, early detection and timely handling of fires are key factors in minimizing these damages. In the past, traditional methods focused on using heat and smoke sensors, which have certain latency drawbacks and are not suitable for some specific environments. Currently, many studies have improved and applied deep learning models such as Faster R-CNN, VGG, and their variants, as well as the YOLO

family. Among them, YOLOv10, the latest version of the YOLO family, has proven effective in real-time object detection. Although it has been researched and applied in various fields, YOLOv10 is still in the development and improvement phase.

Advanced convolutional deep learning models in object detection, such as Faster R-CNN and the newer variants of the VGG model, have been shown to achieve high accuracy in fire detection [1–3]. However, their main drawback is relatively slow processing and prediction times, making these models unsuitable for real-time applications. YOLO (You Only Look Once), especially the latest versions like YOLOv10, stands out for its fast processing speed, making it a strong candidate for low-latency applications [4]. However, YOLOv10 faces limitations in accuracy when compared to other modern CNN models like Faster R-CNN and VGG versions.

YOLOv10 is a modern deep learning model with strong object detection capabilities and has been widely applied in various fields. It has proven highly effective for real-time wildfire detection under conditions where the image quality from satellites or unmanned devices is adequate. However, when using satellite images with insufficient resolution, detecting wildfires at a very early stage can be challenging. Factors such as low light, cloud cover, or adverse weather can affect YOLOv10's detection accuracy [5]. In practice, these images often have poor quality, resulting in suboptimal performance and making practical application difficult. On the other hand, fire detection and warning in buildings or factories are often easier to implement due to easy integration with existing cameras. In such environments, with numerous devices, complex structures, and interference sources, YOLOv10 can effectively detect fires based on its ability to detect multiple objects within a single frame [4,6]. However, images obtained from cameras are also affected by environmental factors such as brightness, visual noise, sparks from production, and flickering lights, posing significant challenges. By applying data augmentation techniques, we created a fire image dataset specifically tailored for buildings, factories, and warehouses suitable for the fine-tuned YOLOv10 model to enhance fire detection accuracy while maintaining superior detection speed, thereby improving the efficiency of real-time fire warning systems.

To leverage the advantages of YOLOv10 in real-time applications, this study focuses on enhancing the accuracy of YOLOv10 through model fine-tuning, aiming to optimize the deep learning model by adjusting its critical components and parameters based on a specific dataset. Additionally, this research applies appropriate data augmentation techniques to expand and enrich the training dataset, thereby enabling the model to learn better features and improve overall accuracy.

The main contributions of our study include: (i) creating a diverse fire image dataset with variations in size, shape, color, brightness, contrast, dimension, rotation angle, and resolution, specifically for buildings, factories, and warehouses; (ii) modifying the attention mechanism by freezing certain initial layers of the model to retain general features learned from the initial dataset; and (iii) changing the optimization algorithm from Adam to Stochastic Gradient Descent (SGD) with momentum. This adjustment is more suitable for fire detection applications in buildings, factories, and warehouses due to the high generalization capability of the dataset, especially in the initial layers when the freezing technique is applied using the Squeeze and Excitation (SE) attention mechanism to improve the detection accuracy.

The remainder of the paper is organized as follows. In Section 2, we discuss modern fire detection methods concerning accuracy and real-time assurance. Section 3 presents the method of constructing a fire dataset for buildings, factories and warehouses (hereafter referred to as the Fire Indoor dataset). Section 4 presents the training results, evaluation, and discussion. Finally, conclusions and recommendations are provided in Section 5.

2 Related Works

The issue of fire detection in buildings and production areas using deep learning Convolutional Neural Network (CNN) models has been extensively studied, with various approaches leveraging models such as Faster R-CNN, VGG versions, and models of the YOLO family. In this section, we analyze several recent notable works related to these models and similar CNN architectures to enhance fire detection effectiveness.

The study [7] developed a real-time fire detection model using the Faster R-CNN (Faster Region-based Convolutional Neural Network) model [8], an advanced method to improve the performance and speed of object detection in images through the use of a Region Proposal Network (RPN). The system is designed to operate in real-time, utilizing a webcam and a deep learning model based on Faster R-CNN for fire detection, making it suitable for applications that require quick response, such as automatic fire fighting systems. The research shows that the fire detection model based on Faster R-CNN can be effectively used for real-time fire detection with high accuracy. However, this model was trained with only 1000 images. Although it achieved high accuracy, the limited data scope could reduce the model's effectiveness when deployed in more diverse real-world conditions, and it might have faced overfitting issues, given that the accuracy results reached 99%.

In [9], the authors modified the SqueezeNet architecture for energy-efficient fire detection in video surveillance, balancing accuracy and computational efficiency. It achieves this by minimizing the use of densely connected layers and employing smaller convolutional kernels, which reduces computational requirements. Although the study emphasizes energy efficiency, there is a trade-off in detection speed and accuracy in complex scenarios, which is not discussed. Study [10] focused on developing a fire detection algorithm using a 3D CNN approach, specifically aimed at enhancing fire detection capabilities in outdoor environments. The authors use a dataset comprising 124 videos, categorized into three types negative videos (no fire or smoke), smoke videos, and fire videos. This classification helps train the model to distinguish between different scenarios. The 3D CNN architecture was chosen for its ability to analyze spatial and temporal features in video data, making it highly suitable for detecting fires in video footage. Experimental results show the effectiveness of the 3D CNN approach in fire detection, emphasizing the importance of longer training times to achieve higher accuracy. However, the authors used a small dataset for training deep learning models. A limited dataset can reduce the model's ability to generalize well to new, unseen data. The study emphasizes that increasing the number of epochs improves accuracy, with the model achieving 100% accuracy after 50 epochs, indicating an overfitting problem with the proposed model. Study [11] proposed an efficient CNN for Fire Detection (CNN-FD), specifically designed for fire detection. The model is innovative in using multi-path convolutional layers with different filter sizes (1×1, 1×3, 3×1, and 3×3), allowing for the capture and learning of both local and global features from images. The proposed model was trained and tested using a custom dataset. The authors curated and developed a dataset tailored to the specific needs of fire detection, enhancing the model's performance and applicability in real-world scenarios. Experiments show that the performance of the proposed CNN model surpasses that of existing state-of-the-art algorithms such as VGG16, VGG19, Inception v3, and Xception. However, deep convolutional neural networks typically require significant computational resources for training and inference, and the study does not address the computational efficiency of the proposed model, which could be a limitation for deployment in resource-constrained environments.

The study [12] provided a quantitative comparison of two deep learning methods, the YOLOv7 algorithm and CNNs, specifically for detecting hazardous fires from CCTV images. The YOLOv7 algorithm achieved a mean average precision (mAP) of 0.45, compared to a mAP of 0.32 for CNN,

demonstrating its superior performance in this context. However, the study utilized a relatively small dataset of 68 samples, which may limit the generalizability of the findings. The study context did not specify the diversity of the dataset in terms of fire scenarios, environments, or different conditions of the image. This lack of diversity could impact the algorithm's performance in various real-world situations. The TRA-YOLO network, proposed by Xiang et al. [13] integrated a transformer encoder with CNN to enhance the extraction of both global and local features, thereby improving fire detection efficiency in factories. It utilizes an improved ACK-Res2Net module and adaptive spatial feature fusion to suppress noise and enhance critical information, thus boosting the model's anti-interference performance and reducing false alarms. Experimental results show that TRA-YOLO achieves a 4.1% increase in accuracy compared to YOLOv5. However, the proposed network in this study struggles to accurately distinguish between actual fire events and other visually similar patterns, leading to potential false alarms. Additionally, the model may not be robust enough to detect and accurately classify flames of varying shapes. Song et al. proposed a real-time fire detection method based on the YOLOv9 model [14]. The study focuses on optimizing the algorithm's principles based on programmable gradient information and generalized ELAN, enhancing object detection capabilities to demonstrate YOLOv9's effectiveness in fire detection. However, the authors focus more on its advantages and practical applications without comparative evaluation against related models. An improved YOLOv8n model is proposed in [15] for detecting fire and smoke in factories. This proposal combines ConNeXtV2 and other enhancements, achieving over 95% accuracy on key metrics, thus meeting real-time monitoring requirements. Although the improved algorithm meets real-time monitoring requirements, the increased complexity may pose challenges for deployment, especially in resource-constrained environments.

These studies demonstrate advancements in fire detection using deep learning models based on CNNs, highlighting improvements in accuracy, real-time processing, and resource efficiency. However, challenges remain due to the highly diverse nature of fire scenarios in real-world applications, which future research must address.

3 Methodology

3.1 Constructing the Dataset

This section details the process of building a dataset focused on indoor fire detection based on publicly available fire detection datasets to train a deep learning model for fire detection. These datasets are sourced from Roboflow's University repository and include images captured under various conditions, perspectives, lighting, and spatial configurations. To enhance the model's ability to generalize different indoor fire detection scenarios, the dataset was constructed through the following key steps:

(1) Data Collection: Sources were gathered from publicly available fire detection datasets from the repository of Roboflow's University Projects [16].
(2) Preprocessing: This involved removing EXIF orientation information, performing annotations, adjusting image resolution, and resizing all images to 640 × 640 pixels.
(3) Data Augmentation: Key techniques included horizontal flipping, rotating images 90 degrees clockwise and counterclockwise, cropping with a zoom range from 0% to 20%, random rotations between −15 to +15 degrees, and brightness adjustments from 0% to +20%.
(4) Compilation: All adjusted images were compiled into a dataset called the Fire Indoor dataset, which is publicly available at [17].

This comprehensive data preparation method ensures that the model can effectively recognize and respond to fire incidents in various indoor contexts, which is crucial for practical applications. Details about the data sources are described in Table 1. The first data source contains images of small fires inside buildings, which can be used to train models for detecting small fires in indoor environments. We collected 755 images of small fires with 1208 annotations, averaging 1.6 annotations per image. The image resolutions range from 0.41 to 0.92 megapixels. The second data source includes images of larger fires, both indoors and outdoors, useful for training models to detect large-scale fires and for diversifying environmental recognition. We collected 568 images of large fires with 965 annotations, averaging 1.7 annotations per image, with resolutions ranging from 0.07 to 17.92 megapixels. The third data source contains images of fires of various sizes, both indoors, in production areas, and outdoors, enhancing the dataset's diversity for training models. We collected 1398 images of fires of varying sizes with 3495 annotations, averaging 2.5 annotations per image, and image resolutions ranging from 0.05 to 1.27 megapixels. The fourth data source features a variety of fire images of different shapes and sizes, both indoors, in production areas, and outdoors, suitable for diverse fire detection and classification tasks. We collected 2790 images of fires of various shapes and sizes with 6138 annotations, averaging 2.2 annotations per image, with image resolutions ranging from 0.23 to 0.41 megapixels. The final data source contains images of fires inside buildings and production areas in various hidden or hard-to-detect locations, which can be used to train models for complex fire detection scenarios. We collected 489 images of indoor fires in various locations with 586 annotations, averaging 1.2 annotations per image, with resolutions ranging from 0.13 to 3.69 megapixels.

Table 1: Fire image database

Number of images	Number of annotations (Average per each image)	Image aize (mp)	Description
755	1208 (1.6)	0.41 to 0.92	Images of small fires inside buildings
568	965 (1.7)	0.07 to 17.92	Images of large fires inside buildings and outdoors
1398	3495 (2.5)	0.05 to 1.27	Images of fires of various shapes and sizes, both indoors, factories, and outdoors
2790	6138 (2.2)	0.23 to 0.41	Images of fires of various shapes and sizes, both indoors, in factories, warehouses, and outdoors
489	586 (1.2)	0.13 to 3.69	Images of fires in buildings, factories, warehouses in hidden or hard-to-detect locations

Each annotation file for an image provides five information fields: class_index, x_coordinate, y_coordinate, width, and height, as listed in Table 2. The 'class_index' is an integer that represents the class of the object corresponding to the bounding box in the image. It is used to identify the type of object detected in the image. The 'x_coordinate' is a floating-point number in the range [0, 1], representing the x-coordinate of the center of the bounding box, indicating the relative horizontal position within the image. This value is normalized from 0 to 1, where 0 represents the left edge of the

image and 1 represents the right edge. Normalization ensures the model operates consistently across images of different sizes. The 'y_coordinate' is a floating-point number in the range [0, 1], representing the y-coordinate of the center of the bounding box, indicating the relative vertical position within the image. This value is also normalized from 0 to 1, where 0 is the top edge of the image and 1 is the bottom edge. Like the x-coordinate, this normalization allows the model to effectively process images with varying scales. The 'width' is a floating-point number in the range [0, 1], indicating the relative width of the bounding box in the image. This value is normalized from 0 (no width) to 1 (maximum width equal to the width of the entire image). Normalization helps represent the bounding box size regardless of the actual size of the image. The 'height' is a floating-point number in the range [0, 1], indicating the relative height of the bounding box in the image. This value is normalized from 0 (no height) to 1 (maximum height equal to the height of the entire image). Similar to width, normalizing the height helps represent the bounding box size consistently across images of different sizes. These normalized values ensure that the model can handle images of varying dimensions efficiently, providing a uniform representation for object detection tasks.

Table 2: Details of annotations

Field	Format/Possible value	Description
<class_index>	Integer	The class index of the object corresponds to the bounding box
<x_coordinate>	[0, 1]	The x-coordinate of the center point represents the relative position of the bounding box within the image
<y_coordinate>	[0, 1]	The y-coordinate of the center point represents the relative position of the bounding box within the image
<width>	[0, 1]	Indicates the relative width of the bounding box within the image
<height>	[0, 1]	Indicates the relative height of the bounding box within the image

Table 3 summarizes the statistics of fire images in the dataset. The total number of fire images in the dataset is 6000, including 5075 indoor images and 925 outdoor images. The dataset contains 5899 fire objects, with 5040 indoor objects and 859 outdoor objects. The 'disturbances' field indicates the number of images containing non-fire elements that complicate fire and non-fire classification, such as lighting or objects with shapes similar to flames. There are 35 disturbance images in indoor environments and 66 disturbance images in outdoor settings, totaling 101 disturbance images in the dataset, which accounts for nearly 2% of the total images. For this study, the overall dataset is divided into an 80:15:5 ratio, comprising 4800 samples in the training set, 900 samples in the validation set, and 300 samples in the test set.

3.2 Fine-Tuning the YOLOv10

The key components impacting the YOLOv10 model in the fine-tuning approach of this study are presented in two main blocks, as illustrated in Fig. 1. The left block represents the overall model structure, while the right block shows the enhanced modules in the YOLOv10 model. Specifically, these include:

(1) Backbone: This is a convolutional neural network responsible for extracting features from the input images, with the output being features extracted at multiple resolution levels. In YOLOv10, this component uses the Cross Stage Partial Network (CSPNet) architecture to improve gradient flow and reduce computational redundancy. CSPNet divides the feature extraction process into multiple processing groups. In each group, the feature map is split into two parts for processing and then merged back together [18]. Additionally, the Spatial-Channel Decoupled Downsampling (SCDD) technique is applied to enhance processing by splitting it into two stages: spatial downsampling and depth enhancement. This helps reduce computational costs and maximize the retention of important information in the image. In this model, after the first convolution, the SE layer is applied. This layer performs two main operations: (1) Global average pooling is used to squeeze the spatial dimensions of each feature map into a single number. (2) The squeezed descriptors are passed through two fully connected layers with a non-linear activation function to generate per-channel weights. These weights are then used to reweight the feature maps. The output from the SE block is passed through a sequence of Residual Blocks, they use skip connections to alleviate the problem of vanishing gradients. After the residual blocks, the feature maps pass through a depthwise convolution layer. Depthwise convolution applies a separate convolution to each input channel independently, as opposed to standard convolution which combines all input channels. This layer drastically reduces the number of computations needed. The output of the depthwise convolution is passed through a pointwise convolution layer, which is a 1×1 convolution. This layer combines the feature maps from the previous depthwise convolution and allows for learning channel-wise relationships.

(2) Neck: This is a crucial part of the overall network architecture, designed to aggregate input features from different resolution levels and output enhanced and combined features. In YOLOv10, the Path Aggregation Network (PAN) is an improved neural network aimed at enhancing feature aggregation by adding shortcut connections to optimize the flow of information from lower to higher layers.

(3) Head: This part is responsible for object detection based on the aggregated features from the Neck. YOLOv10 leverages the advantages of two labeling methods, OneToMany and OneToOne, to eliminate the Non-Maximum Suppression (NMS) mechanism, thereby reducing computational costs during prediction [4]. The OneToMany network is used during training, while the OneToOne network is used during testing. This reduces dependence on post-processing algorithms and enhances the real-time performance of the model.

Table 3: Statistics of fire images in the dataset

Scenario	Objects		Images
	Fire	*Disturbances*	
Indoor	5040	35	5075
Outdoor	859	66	925
Total	5899	101	6000

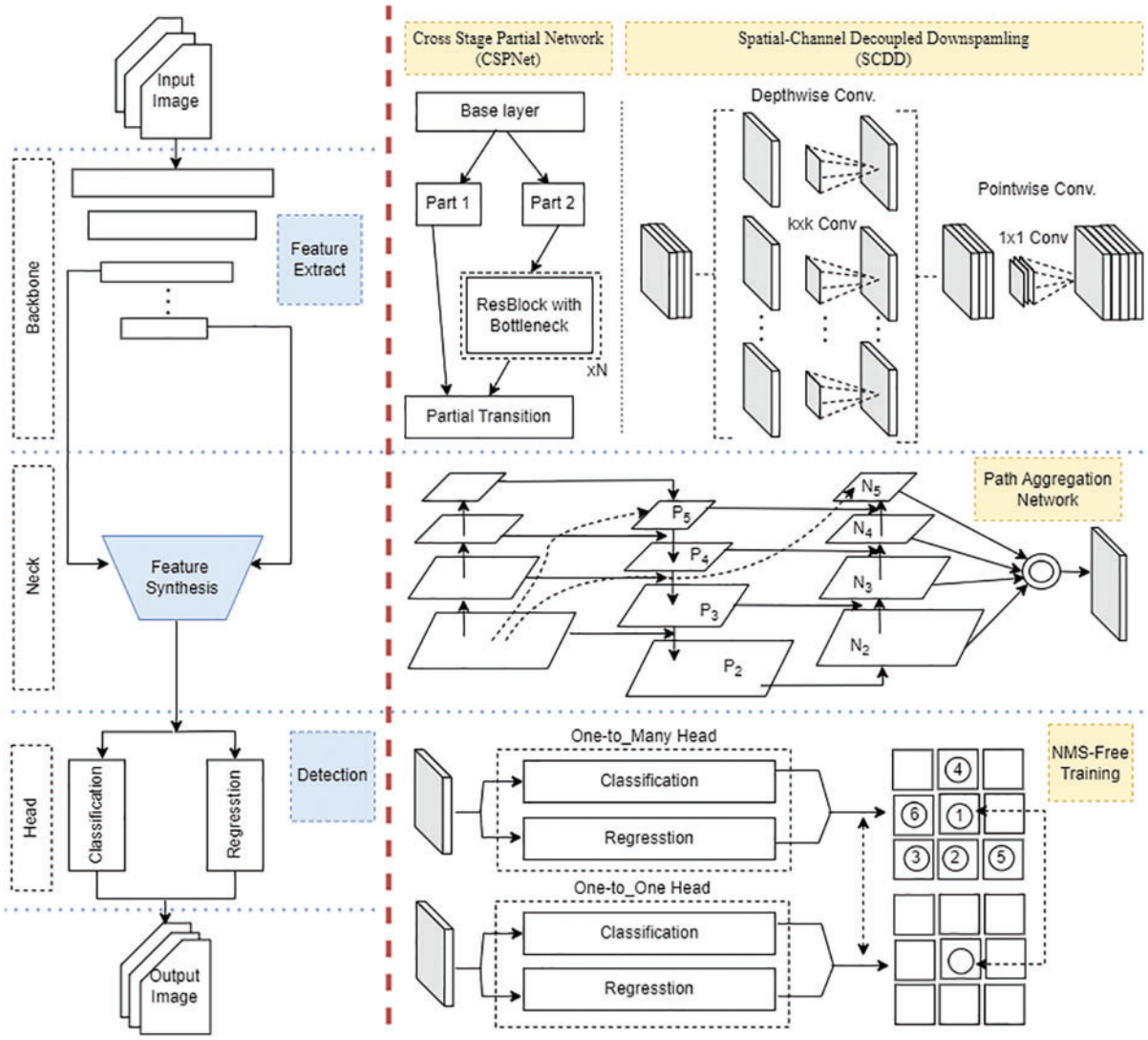

Figure 1: YOLOv10 structure (left) and Improved modules (right)

These components and enhancements provide a robust structure for YOLOv10, improving feature extraction, aggregation, and prediction efficiency, making it suitable for real-time fire detection applications.

SE attention and Partial Self (PS) attention are mechanisms designed to enhance the performance of neural networks by focusing on important features in the data, but they differ significantly in their approach and application. SE attention recalibrates feature responses channel-wise by globally aggregating spatial information and applying learned scaling factors to emphasize more relevant features, particularly in the context of CNNs for tasks like image classification. SE Attention is lightweight, computationally efficient, and excels at enhancing feature discrimination at the channel level without incurring significant costs [19]. On the other hand, PS attention is a variant of the traditional self attention mechanism that selectively attends to a subset of input elements instead of all possible pairs. This approach aims to reduce the computational burden associated with full

self-attention, which requires computing pairwise attention scores across all elements in a sequence. PS Attention is particularly useful in tasks involving long sequences, such as in Natural Language Processing (NLP), where full self-attention would be very costly. It strikes a balance between capturing essential dependencies in the data and maintaining computational efficiency, though at the cost of potentially missing some interactions compared to full self-attention [20]. In YOLOv10, the authors use the PS Attention technique to replace self attention used in earlier versions of YOLO to reduce computational costs. However, this substitution leads to a decrease in accuracy, while fire detection applications in buildings and production areas require high accuracy to avoid errors that could result in significant damage to people and property. To fine-tune the model,in this study, we use the SE attention technique instead of PS attention to improve accuracy while still ensuring real-time fire detection performance.

In fire prediction problems for indoor environments or factories, using an appropriate optimization algorithm is crucial to ensure that the model can accurately predict rare and potentially hazardous events. Study [21] highlights that Adam can encounter convergence problems when gradients change unstably or when data is highly random, causing the model to get trapped in local optima. This can be detrimental in problems like fire prediction, where data is often non-homogeneous and highly noisy. In contrast, research shows that SGD with momentum tends to converge more stably and has a better ability to escape local optima. The authors also demonstrate that SGD with momentum often provides better generalization results, which is critical in applications that require high accuracy, such as fire prediction. In the study [22], the authors point out that for some complex and noisy datasets, SGD with momentum often achieves better generalization performance than Adam. Specifically, when conducting experiments on the ResNet model and large image datasets like CIFAR-10 and ImageNet, they found that SGD with momentum outperformed Adam in achieving higher accuracy on the test set. This work also emphasizes that Adam often leads models to perform well on the training set but may not generalize well on the test set, particularly when dealing with highly diverse data. In contrast, although SGD with momentum converges more slowly, it is better at finding general minima, helping the model generalize better on new data. To fine-tune the model, in this study, we use SGD with momentum instead of Adam to improve accuracy while ensuring real-time fire detection performance.

As shown in Table 4, different YOLOv10 models vary in complexity and performance characteristics. YOLOv10m has 498 layers, 16,485,286 parameters, and 16,485,270 gradients. YOLOv10n, a lighter variant, includes 385 layers, 2,707,430 parameters, and 2,707,414 gradients. YOLOv10s, which balances between these two, consists of 402 layers, 8,067,126 parameters, and 8,067,110 gradients. The most complex among them is YOLOv10b, with 518 layers, 20,452,566 parameters, and 20,452,550 gradients. All these variants are optimized using the SGD with momentum (SGDM) optimizer and the SE Attention technique. Each model variant balances layer depth and parameter count to address different performance and efficiency requirements.

Table 4: YOLOv10 models specifications and fine-tuning

	Layers	Parameters	Gradients	Optimizer	Attention
YOLOv10n	385	2,707,430	2,707,414	SGDM	SE
YOLOv10s	402	8,067,126	8,067,110	SGDM	SE
YOLOv10m	498	16,485,286	16,485,270	SGDM	SE
YOLOv10b	518	20,452,566	20,452,550	SGDM	SE

3.3 Evaluation Metrics

To evaluate object detection performance, Average Precision (AP) is commonly used through several metrics such as Intersection over Union (IoU), Precision, and Recall. According to [23], these metrics are defined by Eqs. (1)–(3). According to Eq. (1), IoU assesses the overlap between the predicted bounding box (pred) and the ground truth box (gt). Precision (P) measures the accuracy of the predictions, while Recall (R) evaluates the detector's ability to identify all gt instances. According to Eqs. (2) and (3), Precision is the ratio of the number of True Positives (TP) to the total number of positives, including TP and False Positives (FP), while Recall is the ratio of the number of True Positives (TP) to the total number of Ground Truth Positives, including TP and False Negatives (FN). In a fire detection system, high Precision means that most fire alerts are accurate, with few false alarms, while high Recall means the system detects most fires, even though there may be some false positives. The F1 score, calculated according to Eq. (4), is a combined metric of Precision and Recall that provides a comprehensive measure of the model's performance. The F1 score is particularly useful when there is an imbalance between Precision and Recall, helping to balance between minimizing false alarms and detecting true positive cases. In a fire detection system, a high F1 score indicates that the system can both detect many fires (high Recall) and limit the number of false alarms (high Precision).

$$IoU = \frac{Area\ of\ Overlap}{Area\ of\ Union} = \frac{Area_{pred} \cap Area_{gt}}{Area_{pred} \cup Area_{gt}} \tag{1}$$

$$Precision = \frac{TP}{TP + FP} \tag{2}$$

$$Recall = \frac{TP}{TP + FN} \tag{3}$$

$$F1 - score = \frac{2\,(Precision \times Recall)}{Precision + Recall} \tag{4}$$

Average Precision (AP) is the area under the Precision-Recall curve (PR AUC), which is a graph that represents the relationship between Precision and Recall for different prediction thresholds. The general formula to compute AP for an object class *iii* can be expressed as Eq. (5).

$$AP_i = \int_0^1 Precision\,(r)\,dr \tag{5}$$

where r records each Recall level from 0 to 1 at intervals of 0.1, i.e., at points $\{0, 0.1, 0.2, \ldots, 1.0\}$. For each of these points, $Precision(r)$ represents the highest Precision value achieved for any Recall level greater than or equal to r.

In object detection, there can be multiple classes of objects that need to be detected. The mean Average Precision (mAP) metric is used to compute the AP value for each class and then calculate the average of these APs. This metric provides a comprehensive assessment of the model's performance across all classes, accounting for the varying levels of difficulty in detecting different objects. According to [24], mAP is described in Eq. (6).

$$mAP = \frac{1}{N} \sum_{i=1}^{N} AP_i \tag{6}$$

where N represents the number of object classes, and AP_i is the Average Precision for class i.

In addition to evaluating the model's accuracy metrics, this study also assesses model efficiency through the metric of inference prediction time. This is a crucial metric for real-time applications. It is calculated by measuring the time required for the model to read an image, process it through the model, and generate prediction results. Typically, this time is averaged over a set of input images.

4 Results and Discussion

In addition to selecting the SGD with momentum optimization algorithm and the SE attention technique to improve accuracy and increase generalization, in the experiment, we adjusted several parameters to train the model for 100 epochs. These parameters include a weight decay of 0.0005, an initial learning rate of 0.01, a warmup epoch of 5, a warmup learning rate of 0.001, a warmup momentum of 0.8, a final learning rate of 0.01, a batch size of 16, and an image size of 640. Additionally, we applied a mosaic parameter of 1.0 for the first 90 epochs and the close mosaic for the last 10 epochs, with the momentum set to 0.9. These configurations were applied for training, validation, and testing of all four YOLOv10 models (n, s, m, b) on the Fire Indoor dataset. The training environment was set up on Google Colab, utilizing an Nvidia Tesla L4 GPU. This setup allowed for efficient training of the models, optimizing them for accurate and real-time fire detection performance in indoor settings.

As shown in Table 5, training different YOLOv10 models for 100 epochs reveals differences in accuracy and inference time. In this study's experiments, we used mAP@50 and mAP50–95 calculations according to Eq. (6), corresponding to IoU thresholds of 0.5 and from 0.5 to 0.95 with a step of 0.05.

Table 5: Detection accuracy and prediction time of YOLOv10 model on fire indoor dataset

Fine-tuned models	P	R	Inference predict (ms)	mAP@50	mAP@50–95	F1 score
YOLOv10n	0.863	0.866	10.5	91.1%	60.6%	0.864
YOLOv10s	**0.906**	**0.846**	**10.7**	**91.3%**	**62.1%**	**0.875**
YOLOv10m	0.891	0.864	14.6	91.4%	62.9%	0.877
YOLOv10b	0.905	0.851	20.2	90.9%	62.8%	0.877

The YOLOv10n (nano) model achieved the lowest precision at 0.863, the highest overall recall at 0.866, and the lowest F1 score at 0.864. It achieved the mean average precision at an IoU threshold of 0.5 (mAP@50) of 91.1%, and the inference prediction time of 10.5 ms. The YOLOv10s (small) model showed improved results with a precision of 0.906, recall of 0.846, and F1 score of 0.875, achieving the mAP@50 of 91.3%, and an inference prediction time of 10.7 ms. The YOLOv10m (medium) model recorded a precision of 0.891, recall of 0.864, and an F1 score of 0.877, leading to the highest mAP@50 of 91.4%, and an inference prediction time of 14.6 ms. The largest model, YOLOv10b (balanced), achieved a precision of 0.905, recall of 0.851, and an F1 score of 0.877, with a mAP@50 of 90.9%, and an inference prediction time of 20.2 ms. These findings suggest that the YOLOv10s (small) model offers the best balance between accuracy and inference time, making it the most suitable model for real-time fire detection. The medium and nano models also perform well, with average mAP@50 values fairly close to each other; however, the medium model's inference time is significantly longer than the small and nano models (14.6 ms compared to 10.7 and 10.5 ms, respectively). Meanwhile, the balanced

model, although it has good precision and F1 score, has a lower average mAP@50 and an inference time nearly double that of the smaller models.

Additionally, for a more accurate evaluation when deploying deep learning models in practical applications, the mAP@50–95 metric is often emphasized over mAP@50 because it requires the model not only to correctly detect objects but also to precisely localize them with higher accuracy (at higher IoU thresholds). In Table 5, the mAP@50–95 value for the YOLOv10s (small) model is 62.1%, which is close to the medium (62.9%) and balanced (62.8%) models, while being significantly better than the nano model (60.6%). This result indicates that the small model is very well-suited for tasks that require high accuracy and good generalization capabilities, such as fire prediction in buildings, factories, or warehouses.

Fig. 2 presents various metrics over 100 epochs of training the YOLOv10s (small) model. Charts in the first row show the training loss curves for different components of the model including train/box_om, train/cls_om, train/dfl_om, train/box_oo, and train/cls_oo. Each of these charts shows a consistent decrease in loss values as training progresses, indicating effective learning and convergence of the model. Charts in the second row display performance metrics including Recall, mean average precision at an IoU threshold of 0.5 (mAP50), mean average precision across IoU thresholds from 0.5 to 0.95 (mAP50-95), and Precision. The 'metrics/recall(B)' and 'metrics/precision(B)' charts show a steady increase in recall and precision values, reaching around 0.9, indicating improved detection capability of the model over time. The 'metrics/mAP50(B)' chart shows an upward trend, approaching 0.91, while the 'metrics/mAP50-95(B)' chart also shows consistent growth, although at a slightly lower value (0.62), reflecting the model's performance across different IoU thresholds. Overall, the YOLOv10s model is learning and improving its performance across various metrics, with a clear trend towards better accuracy, recall, and mean average precision as training progresses. This suggests that the model is effectively optimizing its ability to detect and accurately localize objects, which is crucial for high-stakes applications like fire detection in buildings and factories.

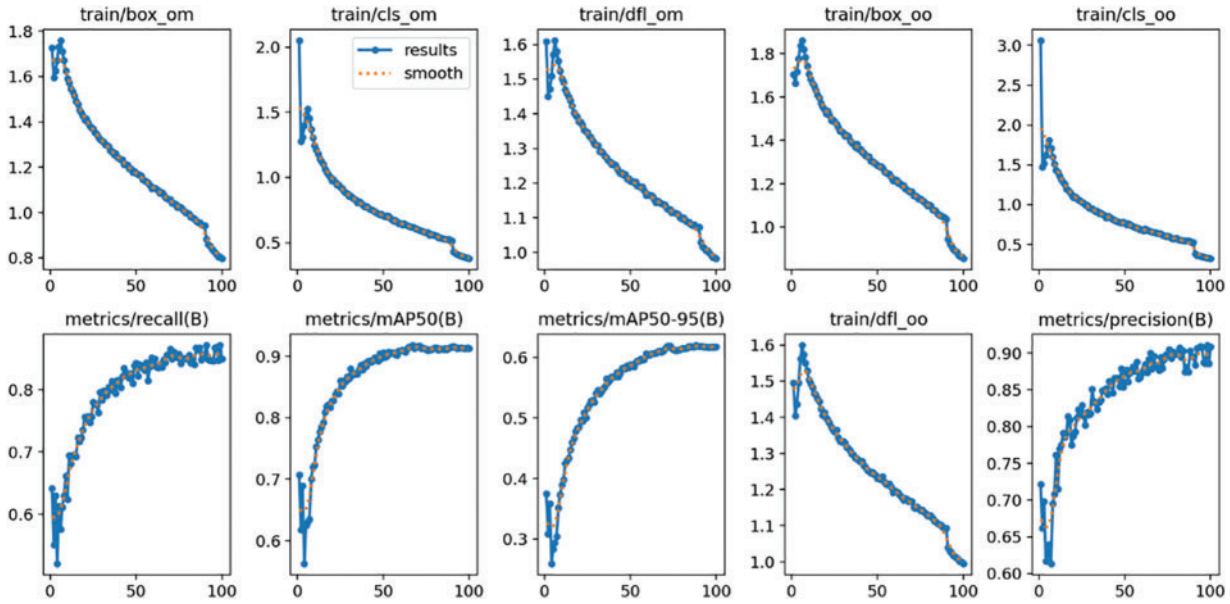

Figure 2: Training results obtained on the fine-tuned YOLOv10s (SGDM optimizer) for the fire indoor dataset

Fig. 3 shows the results of the YOLOv10s model for fire detection in buildings and warehouses after 100 epochs of training, with a batch of predicted output images consisting of 16 fire images in various indoor and warehouse areas, featuring flames of different shapes and sizes. Most predictions have an accuracy of 80% or 90%, with a few having an accuracy between 50% and 80%. This indicates that the YOLOv10s model is capable of effectively detecting fires in diverse environments with varying flame characteristics. The high accuracy rates (mostly around 80%–90%) suggest the model's strong performance in recognizing fire incidents, while the lower accuracy range (50%–80%) in a few cases highlights areas for further refinement to ensure consistent high-quality predictions across all scenarios.

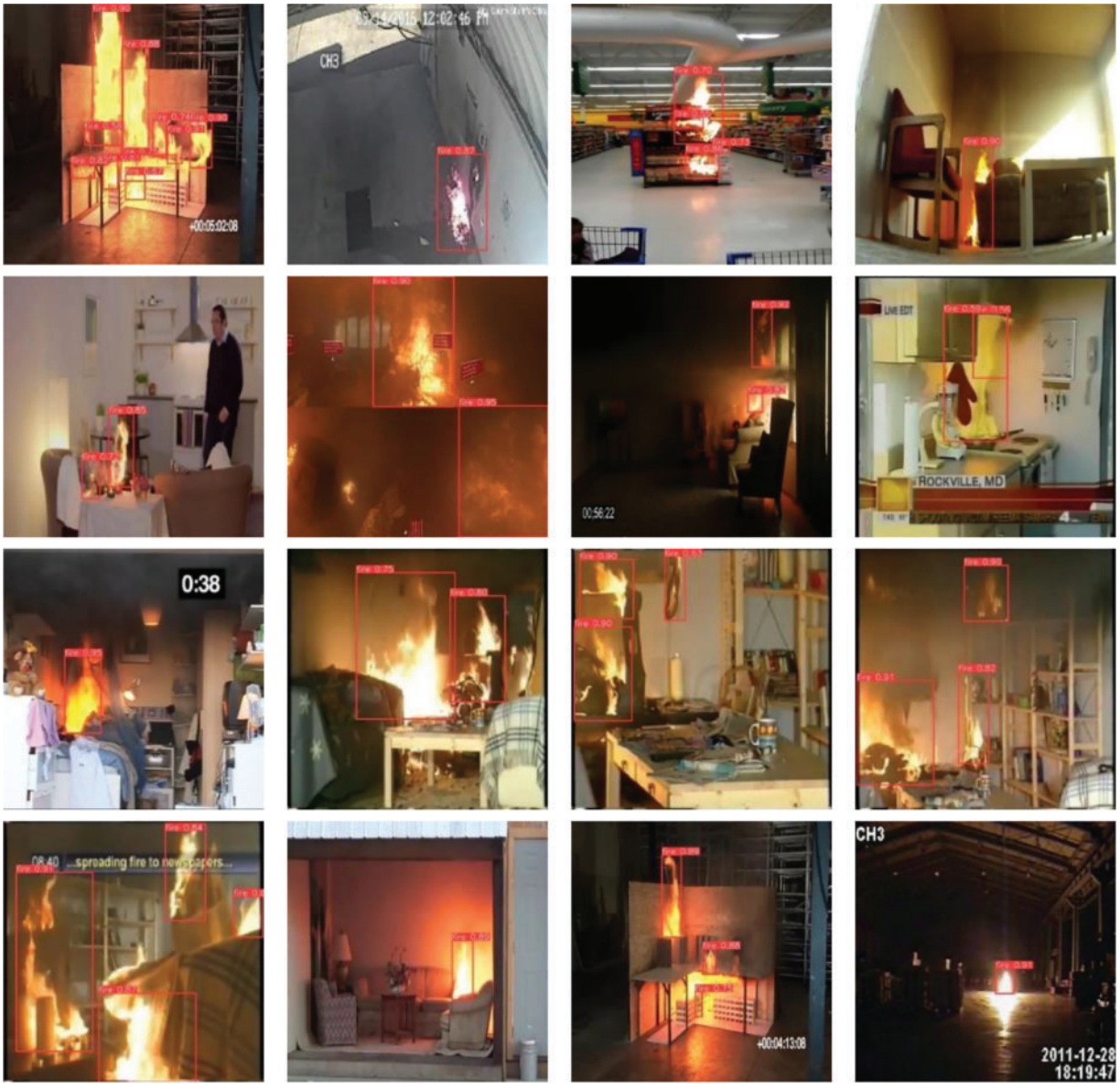

Figure 3: Results of fire indoor detection model—A predicted output image batch

To evaluate the fire detection capability of the fine-tuned YOLOv10s model compared to its baseline version, fine-tuned YOLOv8s model, and Faster R-CNN, we conducted experiments with some parameter and configuration changes. Specifically, for YOLOv10s and YOLOv8s models, we trained for 100 epochs using the SGD with momentum optimizer and the SE attention mechanism, with other parameters remaining the same as in the previous experiment. For the YOLOv10s baseline version, the parameters were similar, except the optimizer was Adam, and the attention mechanism was PS. Faster R-CNN was trained with the parameters: 5000 iterations, batch size of 16, SGD optimizer, learning rate of 0.01, warmup_iters of 100, and warmup_factor of 0.001 (starting from 0.001). The training environment for all models was set up on Google Colab with an Nvidia Tesla L4 GPU. The experimental results in Table 6 show that Faster R-CNN achieved an average mAP@50 of 85.2%, slightly better than the baseline version (82.1%), and mAP@50–95 of 48.4%, which is slightly lower than the baseline version (48.7%). However, these values are much lower than those of the fine-tuned YOLOv10s (91.3% and 62.1%, respectively). Moreover, the inference time for Faster R-CNN averaged 31.1 ms, which is relatively high compared to the baseline YOLOv10s and YOLOv8s and YOLOv10s fine-tuned models (12.7, 18.5, and 10.7 ms, respectively). Thus, it is evident that the Faster R-CNN model cannot compare in terms of performance, both in detection accuracy and prediction time, with the YOLO models. The fine-tuned models such as YOLOv8s or YOLOv10s not only provides significantly better accuracy metrics but also demonstrates faster inference times, making it a more suitable choice for real-time fire detection applications.

Table 6: Detection accuracy and predict time of the fine-tuned YOLOv10 *vs*. other deep learning models

	Optimizer	P	R	Inference predict (ms)	mAP@ 50	mAP@ 50–95	F1 score
Faster R-CNN		–	–	31.1	85.2%	48.4%	–
Baseline YOLOv10s	ADAM	0.8	0.733	12.7	82.1%	48.7%	0.765
Fine-tuned YOLOv8s	SGDM	0.895	0.883	18.3	92.4%	64.5%	0.862
Fine-tuned YOLOv10s	SGDM	0.906	0.846	10.7	91.3%	62.1%	0.875

Compared to the baseline version, the inference time of the fine-tuned YOLOv10s decreased slightly by 6.5% (10.7 *vs*. 12.7 ms), while the accuracy metrics improved significantly. Specifically, the mAP@50 increased by 10% (91.3 *vs*. 82.1), and the F1 score increased by 12% (0.875 *vs*. 0.765). Notably, the mAP@50–95, which is more stringent and comprehensive, increased significantly by 21.5% (62.1 *vs*. 48.7). This indicates a better trade-off achieved by the fine-tuned YOLOv10s compared to the baseline version, considering both inference time and fire detection accuracy. This improvement ensures that the early fire warning system is more effective in terms of providing accurate alerts and minimizing false alarms while still maintaining real-time performance. In addition, we also apply fine-tuning techniques to YOLOv8s and YOLOv10s. The results in Table 6 show that the mAP@50 and mAP@50–95 metrics obtained from training the YOLOv8s model are both better than those of YOLOv10s (92.4% *vs*. 91.3% and 64.5% *vs*. 62.1%, respectively). Although the Recal metric of YOLOv8s is slightly higher, the Precision metric is lower than that of YOLOv10s, which leads to the

F1 score of YOLOv8s being slightly lower than that of YOLOv10s (0.862 *vs*. 0.875). Additionally, the inference time of YOLOv8s is higher than both the baseline and fine-tuned versions of YOLOv10s (18.3 *vs*. 12.7, and 10.7 ms). Such a model is well-suited for practical deployment in buildings and factories, where timely and accurate fire detection is crucial.

Fig. 4 shows the performance metrics of the YOLOv10s model with the Adam optimizer when running for 100 epochs. According to the results from the 'train/dfl_om' chart in Fig. 4, we can clearly see that the convergence of YOLOv10s when using the Adam optimizer changes rapidly at first (from 2.0 to 1.8), but when approaching epoch 100, the loss value is still quite high (1.405). When compared with the corresponding 'train/dfl_om' chart in Fig. 2, we can see that when using the SGDM optimizer, the convergence changes unsteadily in the first epochs, but then the loss value decreases steadily and reaches a very good value at epoch 100 (0.9824). The same thing happens to other loss metrics such as 'train/box_om', 'train/cls_om', 'train/box_oo', 'train/cls_oo', 'train/dfl_oo'. In addition, to evaluate the convergence process between the two optimizers Adam and SGDM on the fire indoor dataset, we analyze the metrics obtained between Figs. 2 and 4 such as 'metrics/recall(B)', 'metrics/precision(B)', 'metrics/mAP50(B)', 'metrics/mAP50-95(B)'. Specifically, with 'metrics/mAP50(B)' measured when training with the Adam optimizer for 100 epochs in Fig. 4, the average accuracy reached 0.821 at epoch 10. In Fig. 2, when training with the SGDM optimizer, the accuracy reached 0.9 at only epoch 52, then stabilized and increased slightly to 0.912 at epoch 100. This confirms that the change from the Adam optimizer to the SGDM is suitable for the diverse fire indoor dataset created in this study, creating high efficiency in the model training process.

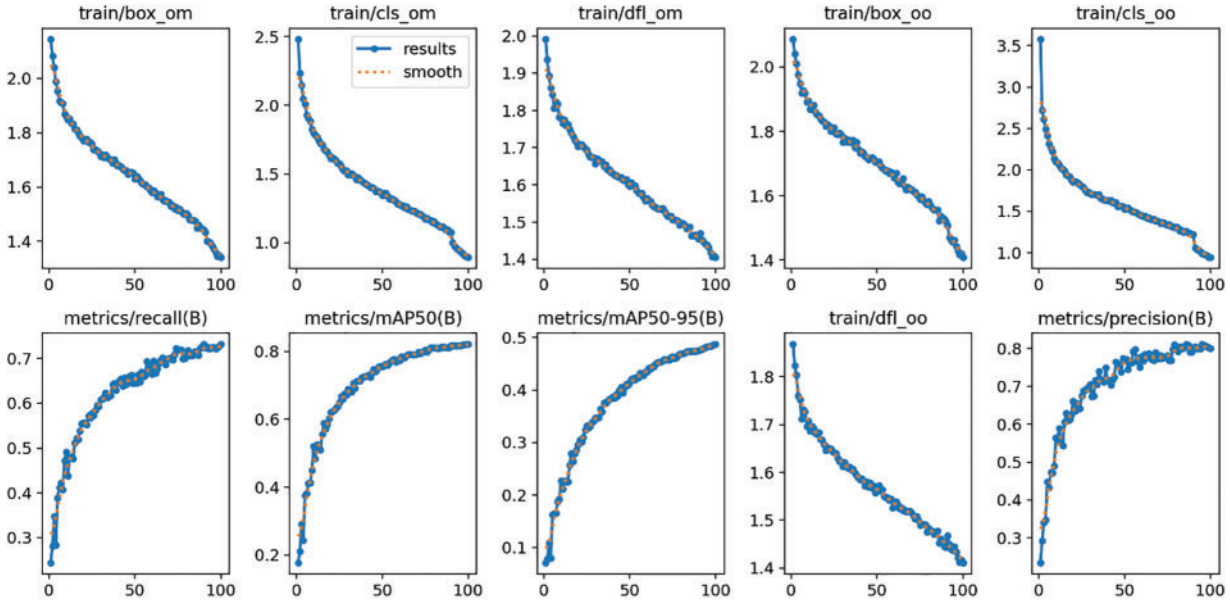

Figure 4: Training results obtained on the baseline YOLOv10s model (Adam optimizer) for fire indoor dataset

5 Conclusion

In this study, we created a fire dataset for buildings and production areas to train fine-tuned deep learning models YOLOv10 (n, s, m, b) and evaluate the performance of different versions of the YOLOv10 model, as well as comparing the YOLOv10s model with the fine-tuned YOLOv8s

and Faster R-CNN models. The experimental results show that the YOLOv10s model is the optimal choice for real-time fire detection applications due to its balance between accuracy and prediction speed. The graphs for metrics such as recall, precision, mAP@50, and mAP@50–95 throughout the training process show stable and consistently increasing trends. This indicates that the model's ability to accurately detect objects improves over time and performs well across different IoU thresholds. While larger models like YOLOv10b achieve slightly higher accuracy, they face significant challenges in prediction time, making them less suitable for systems requiring rapid response. Notably, the mAP@50–95 accuracy metric of YOLOv10s shows very little difference compared to larger models like medium and balanced, demonstrating its well-balanced performance.

However, it is clear that the YOLOv10s model still has some limitations in accuracy when compared to other models like YOLOv10m and YOLOv10b. Therefore, in future research, we plan to combine YOLOv10 and Faster R-CNN to further improve the detection accuracy. Additionally, we will continue deploy YOLOv10 models, particularly the smaller yet effective models like small and nano, on embedded devices and IoT systems to develop real-time fire detection solutions that can be widely applied in practice. shortly. For outdoor environments, we will improve the fire image dataset in different weather conditions (sunlight, rain, fog), different times of day, and different contexts (trees, buildings, vehicles). To do this, additional factors such as smoke flow, reflections, and changing lighting conditions due to weather and time of day need to be addressed. Data augmentation techniques such as rotation, cropping, blurring, brightness changes, and adding environmental noise (rain, smoke, fog) to simulate different conditions when detecting outdoor fires will also be studied. This helps the model learn to detect fires in a variety of environmental conditions. Using outdoor fire datasets to fine-tune the model can help it adapt to different characteristics such as changing lighting, smoke, and weather conditions typical of outdoor spaces. Therefore, applying advanced training techniques such as transfer learning to leverage knowledge from models trained on indoor datasets and then fine-tuning them for outdoor conditions will also be studied.

Acknowledgement: The authors extend their appreciation to the Posts and Telecommunications Institute of Technology (PTIT) in Vietnam for funding this research work.

Funding Statement: The authors wish to express their gratitude to PTIT, Vietnam, for financial support for this research (https://www.ptit.edu.vn/, accessed on 16 October 2024).

Author Contributions: The authors confirm contribution to the paper as follows: study conception and design: Trong Thua Huynh, Du Thang Phu; data collection and curation: Du Thang Phu; software: Du Thang Phu, Hoang Thanh Nguyen; analysis and interpretation of results: Trong Thua Huynh, Hoang Thanh Nguyen; draft manuscript preparation and review: Trong Thua Huynh. All authors reviewed the results and approved the final version of the manuscript.

Availability of Data and Materials: A publicly available Fire Indoor dataset was used for analyzing our model. It can be found at https://universe.roboflow.com/binbin-d1mvq/fire_indoor-3xzzi (accessed on 19 August 2024).

Ethics Approval: Not applicable.

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding the present study.

References

[1] P. Barmpoutis, K. Dimitropoulos, K. Kaza, and N. Grammalidis, "Fire detection from images using Faster R-CNN and multidimensional texture analysis," in *IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, Brighton, UK, 2019, pp. 8301–8305. doi: 10.1109/ICASSP.2019.8682647.

[2] Y. Guerbai and A. Saibi, "DeepFire: Enhanced fire detection using VGG16 convolutional neural networks," *TechRxiv*, Nov. 20, 2023. doi: 10.36227/techrxiv.24548422.v1.

[3] A. Lati, D. Belkebir, L. Bekkari, and K. Nadhir, "Real time fire and smoke detection techniques using improved transfer learning algorithms," in *2023 Int. Conf. Decis. Aid Sci. Appl. (DASA)*, Annaba, Algeria, 2023, pp. 273–277. doi: 10.1109/DASA59624.2023.10286617.

[4] Ultralytics, "YOLOv10-Ultralytics YOLO documents," May 25, 2024. Accessed: Oct. 16, 2024. [Online]. Available: https://docs.ultralytics.com/vi/models/yolov10/

[5] L. A. O. Gonçalves, R. Ghali, and M. A. Akhloufi, "YOLO-based models for smoke and wildfire detection in ground and aerial images," *Fire*, vol. 7, no. 4, Apr. 2024, Art. no. 140. doi: 10.3390/fire7040140.

[6] S. H. Patel, D. H. Patel, N. M. Bankar, N. K. Chaudhari, U. R. Dave and N. K. Prajapati, "Advanced forest fire detection using YOLOv10: A deep learning approach," *J. Electr. Syst. (JES)*, vol. 20, no. 3, pp. 4115–4125, 2024.

[7] H. S. Sucuoğlua, İ. Böğrekcia, and P. Demircioğlu, "Real time fire detection using Faster R-CNN model," *Int. J. 3D Printing Technol. Digit. Ind.*, vol. 3, no. 3, pp. 220–226, 2019.

[8] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, no. 6, pp. 1137–1149, 2017. doi: 10.1109/TPAMI.2016.2577031.

[9] V. R. Balaji, V. S. Shanthini, R. Sri Balaji, and S. L. Stinsha, "Fireguard: Deep CNN video surveillance for efficient fire detection," in *Int. Conf. Sci. Technol. Eng. Manage. (ICSTEM)*, Coimbatore, India, 2024, pp. 1–5. doi: 10.1109/ICSTEM61137.2024.10561206.

[10] D. A. F. Abdurrohman, F. Sthevanie, and K. N. Ramadhani, "Fire detection on video using the 3D CNN method," in *Int. Conf. Data Sci. Appl. (ICoDSA)*, Bandung, Indonesia, 2023, pp. 140–144. doi: 10.1109/ICoDSA58501.2023.10276406.

[11] A. K. Vishwakarma and M. Deshmukh, "CNN-FD: Deep convolutional neural network model for fire detection," in *7th Int. Conf. Comput. Appl. Electr. Eng.-Recent Adv. (CERA)*, Roorkee, India, 2023, pp. 1–6. doi: 10.1109/CERA59325.2023.10455516.

[12] V. P. Praveen Rao and G. Ramkumar, "Effective and precise detection of hazardous fires from CCTV images using YOLOv7 algorithm in comparison with CNN," in *Intell. Comput. Control Eng. Bus. Syst. (ICCEBS)*, Chennai, India, 2023, pp. 1–5. doi: 10.1109/ICCEBS58601.2023.10448517.

[13] S. Xiang, S. Yin, G. Yu, X. Xu, and L. Yu, "Factory fire detection using TRA-YOLO network," in *36th Chin. Control Decis. Conf. (CCDC)*, Xi'an, China, 2024, pp. 3777–3782. doi: 10.1109/CCDC62350.2024.10588005.

[14] L. Song, Y. Wu, and W. Zhang, "Research on fire detection based on the YOLOv9 algorithm," in *IEEE 4th Int. Conf. Electron. Technol., Commun. Inform. (ICETCI)*, Changchun, China, 2024, pp. 1398–1406. doi: 10.1109/ICETCI61221.2024.10594570.

[15] Z. Zhang, L. Tan, and T. L. K. Robert, "An improved fire and smoke detection method based on YOLOv8n for smart factories," *Sensors*, vol. 24, no. 15, 2024, Art. no. 4786. doi: 10.3390/s24154786.

[16] Multi Users, "Roboflow universe projects," Roboflow, Inc., Aug. 10, 2024. Accessed: Oct. 16, 2024. [Online]. Available: https://universe.roboflow.com/roboflow-universe-projects

[17] D. T. Phu, "FIRE_INDOOR computer vision project," distributed by BINBIN, Aug. 19, 2024. Accessed: Oct. 16, 2024. [Online]. Available: https://universe.roboflow.com/binbin-d1mvq/fire_indoor-3xzzi

[18] C. -Y. Wang, H. -Y. M. Liao, Y. -H. Wu, P. -Y. Chen, J. -W. Hsieh and I. -H. Yeh, "CSPNet: A new backbone that can enhance learning capability of CNN," in *IEEE/CVF Conf. Comput. Vis. Pattern Recognit. Workshops (CVPRW)*, 2020, pp. 1571–1580.

[19] J. Hu, L. Shen, S. Albanie, G. Sun, and E. Wu, "Squeeze-and-excitation networks," *Comput. Vis. Pattern Recognit.*, May 2019. doi: 10.48550/arXiv.1709.01507.

[20] C. Zhu *et al.*, "Long-short transformer: Efficient transformers for language and vision," *Comput. Vis. Pattern Recognit.*, Dec. 2021. doi: 10.48550/arXiv.2107.02192.
[21] S. J. Reddi, S. Kale, and S. Kumar, "On the convergence of adam and beyond," in *Int. Conf. Learn. Rep.*, Canada, May 2018. doi: 10.48550/arXiv.1904.09237.
[22] I. Loshchilov and F. Hutter, "On the convergence of adam and beyond," in *Int. Conf. Learn. Rep.*, New Orleans, LA, USA, Jan. 2019. doi: 10.48550/arXiv.1711.05101.
[23] Ultralytics, "Performance metrics deep dive," 2024. Accessed: Oct. 16, 2024. [Online]. Available: https://docs.ultralytics.com/guides/yolo-performance-metrics/
[24] R. Kaur and S. Singh, "A comprehensive review of object detection with deep learning," *Digit. Signal Process.*, vol. 132, Jan. 2023, Art. no. 103812. doi: 10.1016/j.dsp.2022.103812.