[1]Mr. R. Venkadesh, Ankit Kumar, Guduru Venkatesh, Kanyadhra Harish

Flame and Smoke Detection Algorithm Based on ODConvBS-YOLO v5s

Abstract: - This project introduces an improved flame and smoke detection algorithm based on YOLOv5s, incorporating the ODConvBS (Ordinary Convolutional Blocks with Spatial Attention) to enhance the extraction of attentional features from the convolutional kernel. Additionally, the model incorporates Gnconv in the Neck to improve high-order spatial information extraction. The traditional algorithms for flame and smoke detection suffer from issues such as low accuracy, high miss rates, low detection efficiency, and poor performance in detecting small targets, leading to significant losses in fire-related incidents. Using a dataset of flame and smoke, the upgraded YOLOv5s model demonstrated a significant improvement, increase in mean average precision (mAP). The accuracy rate, recall rate, and detection speed also saw substantial enhancements respectively. The proposed algorithm not only addresses the shortcomings of traditional flame and smoke detection methods but also showcases tangible performance improvements, making it a promising solution for real-time and accurate fire detection. The integration of novel components in the model architecture contributes to enhanced feature extraction, leading to better overall detection performance in terms of accuracy, recall, and speed.The project extends its performance capabilities by incorporating advanced detection techniques with YOLO v5x6 and YOLOv8, in which YOLO v5x6 achieved notable metrics of 79.2% mAP, 74% recall, and 80.6% precision.

Keywords: YOLOv5s, object detection, Gnconv, attention mechanism, ODConvBS

1. INTRODUCTION

Fire will inflict significant damage to human life and property in daily activities, as well as do harm to the healthy development of society. However, in the early stages of a fire disaster, the flame is easily extinguished. As a result, by detecting flame and smoke accurately and quickly, the loss caused by the fire may be minimized to sustain normal production. Early flame detection frequently collects flame and smoke data using different temperature sensors, smoke sensors, and photosensitive sensors to assess whether a fire has occurred. However, the installation position and effective range of the sensor, as well as the external light and ambient humidity, will have a significant impact on the detection accuracy of the flame and smoke. The task of object detection is to find and classify all target objects in a picture, which is one of the fundamental tasks in the computer vision field. At the current stage, object detection algorithms are divided into two categories:Twostage [1] and Onestage [2]. The twostage architecture generates pre-selected boxes that may contain objects to be detected and extracted by features and then conducts classification and regression localization. The twostage architecture does not need to generate pre-selected boxes and can extract features directly in the network to predict the classification and location of an object.

With the continuous upgrading of computer vision algorithms and hardware conditions, deep learning-based methods for detecting flame and smoke have surpassed traditional manual methods, and deep learning models can extract more abstract and deeper features from images with more powerful generalization compared to traditional methods. In 2010, Frizzi et al [3] first used a convolutional neural network for the image of flame and smoke detection, which pioneered the feature extraction algorithm of flame and smoke The deep learning-based flame and smoke detection task can be divided into three parts: classification [4] (determining whether the input image contains flame or smoke), detection (identifying whether the image contains flame and smoke and annotating it with an anchor), and segmentation [5] (identifying whether the image contains flame and smoke and annotating its shape).

In 2019, Lin et al [6] developed a joint detection framework by combining Faster R-CNN and 3D CNN, where Faster R-CNN enables smoke localization in static spatial information and 3D CNN [7] achieves the recognition of smoke by combining the information of dynamic spatiotemporal. Compared with the common convolutional detection algorithm of smoke, this method improves significantly in the detection accuracy of smoke. In 2020,

[1,2,3,4]Department Of Computer Science And Engineering , Mahendra Engineering College
Namakkal, Tamilnadu, India.
venkateshguduru0@gmail.com

Li et al [8] used Faster R-CNN [9], R-FCN [10], SSD [11], and YOLOv3 [12] for flame detection, and they found that the CNN-based flame detection model could achieve a better balance of accuracy and detection.
To address the model's high miss detection rate, recursive gated convolution (Gnconv) is introduced into FPN to form a new Gnconv-FPN structure, which improves the model's interaction ability of higher-order information, achieves the same effect as the self-attention mechanism, avoids target information loss, and improves the detection accuracy of small targets even further. To boost the ability to extract features of the model even further, an ultra-lightweight replacement attention mechanism (Shuffle Attention) is included after the FPN structure to integrate all features and perform component feature communication through channel replacement operation. Lastly, the SIOU [27] loss function is used to enhance the training and convergence time of the model by fully accounting for the vector angle between regressions.

2. LITERATURE SURVEY

Due to object detection's close relationship with video analysis and image understanding, it has attracted much research attention in recent years. Traditional object detection [21] methods are built on handcrafted features and shallow trainable architectures. Their performance easily stagnates by constructing complex ensembles which combine multiple low-level image features with high-level context from object detectors and scene classifiers. With the rapid development in deep learning, more powerful tools, which are able to learn semantic, high-level, deeper features, are introduced to address the problems existing in traditional architectures. These models behave differently in network architecture, training strategy and optimization function, etc. In this paper [1], we provide a review on deep learning based object detection frameworks. Our review begins with a brief introduction on the history of deep learning and its representative tool, namely Convolutional Neural Network (CNN) [13,16]. Then we focus on typical generic object detection architectures along with some modifications and useful tricks to improve detection performance further. As distinct specific detection tasks exhibit different characteristics, we also briefly survey several specific tasks, including salient object detection, face detection and pedestrian detection. Experimental analyses are also provided to compare various methods and draw some meaningful conclusions. Finally, several promising directions and tasks are provided to serve as guidelines for future work in both object detection and relevant neural network based learning systems.
Object detection [1,2,9,10] is a fundamental visual recognition problem in computer vision and has been widely studied in the past decades. Visual object detection aims to find objects of certain target classes with precise localization in a given image and assign each object instance a corresponding class label. Due to the tremendous successes of deep learning based image classification, object detection techniques using deep learning have been actively studied in recent years. In this paper [2], we give a comprehensive survey of recent advances in visual object detection with deep learning. By reviewing a large body of recent related work in literature, we systematically analyze the existing object detection frameworks and organize the survey into three major parts: (i) detection components, (ii) learning strategies, and (iii) applications & benchmarks. In the survey, we cover a variety of factors affecting the detection performance in detail, such as detector architectures, feature learning, proposal generation, sampling strategies, etc. Finally, we discuss several future directions to facilitate and spur future research for visual object detection with deep learning. Keywords: Object Detection, Deep Learning, Deep Convolutional Neural Networks.
A video flame detection method based on the multi-feature fusion is presented in this paper [3]. The temporal and spatial characteristics of flames, such as ordinary flame movement and color clues, a flame flickering detection algorithm [8] is incorporated into the scheme to detect fires in color video sequences. An improved Gaussian mixture model method is firstly adopted to extract moving foreground objects from the still background of detection scenes; secondly, detected moving objects are then categorized into candidate and non-candidate flame regions by using a flame color filtering algorithm; finally, a flame flicker identification algorithm based on statistical frequency counting is used to distinguish true flames from fire-like objects in video images. Testing results show that the proposed algorithms are effective, robust and efficient. The processing rate of the flame detection method can achieve 24 fps with image size of 320×240 pixels on a PC with an AMD 2.04 GHz processor.
Fire is the result of chemical reactions taking place between the oxygen and the rest of the atmosphere, which on large scales causes an ecological imbalance by giving out bi-products such as smoke. More than 90 Million cases of fire incidents have been reported since 1990 causing loss of life as well as endangered and vulnerable

resources. The reason most of the time is unknown, but global warming is cited as one of the factors in the case of forest fires. Through this work [4], we develop a multi-disciplinary system that reports the traces of fire and smoke under any isolation. The system uses Computer Vision for analyzing and detecting fire or smoke in real-time through a Deep Learning algorithm [3,4,21] and alerts back upon found.

Fire disaster throughout the globe causes social, environmental, and economical damage, making its early detection and instant reporting essential for saving human lives and properties. Smoke detection plays a key role in early fire detection [14] but majority of the existing methods are limited to either indoor or outdoor surveillance environments, with poor performance for hazy scenarios. In this paper [5], we present a Convolutional Neural Network (CNN)-based smoke detection and segmentation framework for both clear and hazy environments. Unlike existing methods, we employ an efficient CNN architecture, termed EfficientNet, for smoke detection with better accuracy. We also segment the smoke regions using DeepLabv3+, which is supported by effective encoders and decoders along with a pixel-wise classifier for optimum localization. Our smoke detection results evince a noticeable gain up to 3% in accuracy and a decrease of 0.46% in False Alarm Rate (FAR), while segmentation reports a significant increase of 2% and 1% in global accuracy and mean Intersection over Union (IoU) scores, respectively. This makes our method a best fit for smoke detection and segmentation in real-world surveillance settings.

3. METHODOLOGY

i) Proposed Work:

The project proposes the detection algorithm of flame and smoke based on ODConvBS in YOLOv5s [25,26,27]. Firstly, the ordinary convolutional blocks in the backbone network of YOLOv5s are replaced with ODConvBS to achieve the extraction of attentional features from the convolutional kernel. Secondly, Gnconv is introduced into Neck to improve the high-order spatial information extraction ability of the model.As also, the project explores the application of YOLO v8 and YOLO v5x6 models to further enhance the performance of flame and smoke detection, aiming for more accurate final predictionsin which YOLO v5x6 achieved notable metrics of 79.2% mAP, 74% recall, and 80.6% precision. These advanced models bring additional capabilities and improvements in object detection, contributing to the overall efficacy of the detection algorithm. Additionally, it added envisions the development of a user-friendly front end using the Flask framework, facilitating user testing and interaction.

ii) System Architecture:

A complete flame and smoke detection model is built on the computer as shown in Figure 1. The input flame and smoke images first go through feature extraction based on the ODConvBS[25,26,27] backbone network. At the end of the backbone network, the SPPF module, which is faster, is used to unify the scale of the feature maps extracted by the backbone network and improve the accuracy of the feature extraction. The feature maps are then sent to the neck network (Gnconv-FPN) for feature processing and fusion, enabling the interaction of high-order spatial information in the feature maps and achieving the effect of self-attention feature extraction. At the end of the neck network, the SA module is used to promote information fusion between different groups. Finally, the information is sent to the head network to complete the object detection. The improved network model structure is shown in Figure 1.

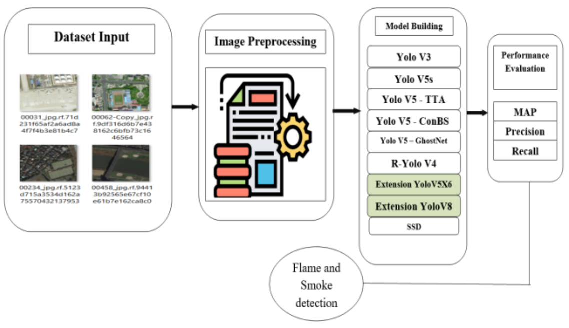

Dataset Input
00031_jpg.rf.71d231f65af2a6ad8a4f7f4b3e81b4c7
00062-Copy_jpg.rf.9df316d6b7e438162c6bfb73c1646564
00234_jpg.rf.5123d715a3534d162a75570432137953
00458_jpg.rf.94413b92565e67cf10e61b7e162ca8c0
Image Preprocessing
Model Building
Yolo V3
Yolo V5s
Yolo V5 - TTA
Yolo V5 - ConBS
Yolo V5 – GhostNet
R-Yolo V4
Extension YoloV5X6
Extension YoloV8
SSD
Performance Evaluation
MAP
Precision
Recall
Flame and Smoke detection

Fig 1Proposed Architecture

iii) Dataset collection:

Due to the small number of images, single scene and low resolution in the public of flame and smoke dataset, it is not conducive to improving the generalization ability of the model. To reflect the improved model generalization and small target detection ability, it is necessary to further improve the diversity of datasets. Therefore, this paper [20] crawls the flame and smoke images on the network through crawlers and then labels the images into data sets to train and evaluate the model. The dataset of 4998 photos was separated into training, validation, and test sets in the ratio of 8:1:1, covering a range of flame smoke scenes, following the study topic of this work.

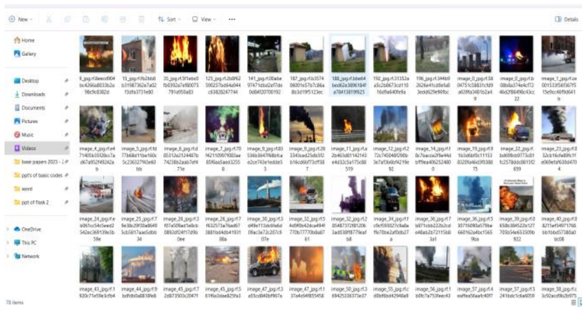


Fig 2 Dataset images

iv) Image Processing:

Image processing plays a pivotal role in object detection within autonomous driving systems, encompassing several key steps. The initial phase involves converting the input image into a blob object, optimizing it for subsequent analysis and manipulation. Following this, the classes of objects to be detected are defined, delineating the specific categories that the algorithm aims to identify. Simultaneously, bounding boxes are declared, outlining the regions of interest within the image where objects are expected to be located. The processed data is then converted into a NumPy array, a critical step for efficient numerical computation and analysis.

The subsequent stage involves loading a pre-trained model, leveraging existing knowledge from extensive datasets. This includes reading the network layers of the pre-trained model, containing learned features and parameters vital for accurate object detection. Additionally, output layers are extracted, providing final predictions and enabling effective object discernment and classification.

Further, in the image processing pipeline, the image and annotation file are appended, ensuring comprehensive information for subsequent analysis. The color space is adjusted by converting from BGR to RGB, and a mask is created to highlight relevant features. Finally, the image is resized, optimizing it for further processing and analysis. This comprehensive image processing workflow establishes a solid foundation for robust and accurate object detection in the dynamic context of autonomous driving systems, contributing to enhanced safety and decision-making capabilities on the road.

v) Data Augmentation:

Data augmentation [25,26] is a fundamental technique in enhancing the diversity and robustness of training datasets for machine learning models, particularly in the context of image processing and computer vision. The process involves three key transformations to augment the original dataset: randomizing the image, rotating the image, and transforming the image.

Randomizing the image introduces variability by applying random modifications, such as changes in brightness, contrast, or color saturation. This stochastic approach helps the model generalize better to unseen data and diverse environmental conditions.

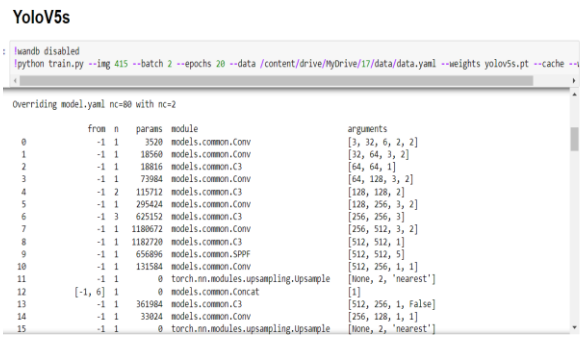
Rotating the image involves varying the orientation of the original image by different degrees. This augmentation technique aids in teaching the model to recognize objects from different perspectives, simulating variations in real-world scenarios.

Transforming the image includes geometric transformations such as scaling, shearing, or flipping. These alterations enrich the dataset by introducing distortions that mimic real-world variations in object appearance and orientation.

By employing these data augmentation techniques, the training dataset becomes more comprehensive, allowing the model to learn robust features and patterns. This, in turn, improves the model's ability to generalize and perform effectively on diverse and challenging test scenarios. Data augmentation serves as a crucial tool in mitigating overfitting, enhancing model performance, and promoting the overall reliability of machine learning models, especially in applications like image recognition for autonomous driving systems.

vi) Algorithms:

YOLOv5s (You only Look Once) is a small and efficient variant of the YOLO object detection algorithm, chosen for its balance between speed and accuracy. It directly predicts bounding boxes and class probabilities, making it suitable for real-time flame and smoke detection [17].

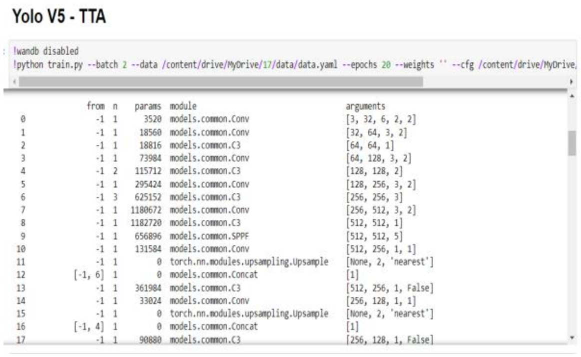


YoloV5s

```
!wandb disabled
!python train.py --img 415 --batch 2 --epochs 20 --data /content/drive/MyDrive/17/data/data.yaml --weights yolov5s.pt --cache --

Overriding model.yaml nc=80 with nc=2

                 from  n    params  module                                  arguments
  0                -1  1      3520  models.common.Conv                      [3, 32, 6, 2, 2]
  1                -1  1     18560  models.common.Conv                      [32, 64, 3, 2]
  2                -1  1     18816  models.common.C3                        [64, 64, 1]
  3                -1  1     73984  models.common.Conv                      [64, 128, 3, 2]
  4                -1  2    115712  models.common.C3                        [128, 128, 2]
  5                -1  1    295424  models.common.Conv                      [128, 256, 3, 2]
  6                -1  3    625152  models.common.C3                        [256, 256, 3]
  7                -1  1   1180672  models.common.Conv                      [256, 512, 3, 2]
  8                -1  1   1182720  models.common.C3                        [512, 512, 1]
  9                -1  1    656896  models.common.SPPF                      [512, 512, 5]
 10                -1  1    131584  models.common.Conv                      [512, 256, 1, 1]
 11                -1  1         0  torch.nn.modules.upsampling.Upsample    [None, 2, 'nearest']
 12           [-1, 6]  1         0  models.common.Concat                    [1]
 13                -1  1    361984  models.common.C3                        [512, 256, 1, False]
 14                -1  1     33024  models.common.Conv                      [256, 128, 1, 1]
 15                -1  1         0  torch.nn.modules.upsampling.Upsample    [None, 2, 'nearest']
```

Fig 3 YOLOV5s

YOLOv5 – TTA (test-time augmentation) extends YOLOv5 by incorporating test-time augmentation during inference. This technique enhances the model's robustness and accuracy by applying various transformations to input images, contributing to more reliable flame and smoke detection under diverse conditions in real-time scenarios.

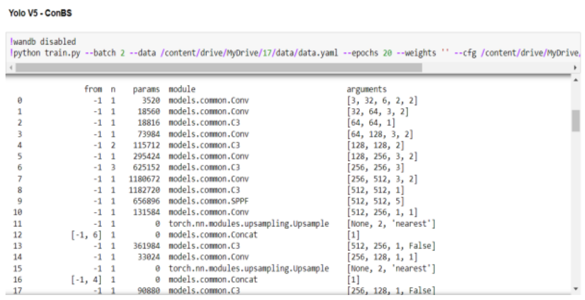


Yolo V5 - TTA

```
!wandb disabled
!python train.py --batch 2 --data /content/drive/MyDrive/17/data/data.yaml --epochs 20 --weights '' --cfg /content/drive/MyDrive,

                 from  n    params  module                                  arguments
  0                -1  1      3520  models.common.Conv                      [3, 32, 6, 2, 2]
  1                -1  1     18560  models.common.Conv                      [32, 64, 3, 2]
  2                -1  1     18816  models.common.C3                        [64, 64, 1]
  3                -1  1     73984  models.common.Conv                      [64, 128, 3, 2]
  4                -1  2    115712  models.common.C3                        [128, 128, 2]
  5                -1  1    295424  models.common.Conv                      [128, 256, 3, 2]
  6                -1  3    625152  models.common.C3                        [256, 256, 3]
  7                -1  1   1180672  models.common.Conv                      [256, 512, 3, 2]
  8                -1  1   1182720  models.common.C3                        [512, 512, 1]
  9                -1  1    656896  models.common.SPPF                      [512, 512, 5]
 10                -1  1    131584  models.common.Conv                      [512, 256, 1, 1]
 11                -1  1         0  torch.nn.modules.upsampling.Upsample    [None, 2, 'nearest']
 12           [-1, 6]  1         0  models.common.Concat                    [1]
 13                -1  1    361984  models.common.C3                        [512, 256, 1, False]
 14                -1  1     33024  models.common.Conv                      [256, 128, 1, 1]
 15                -1  1         0  torch.nn.modules.upsampling.Upsample    [None, 2, 'nearest']
 16           [-1, 4]  1         0  models.common.Concat                    [1]
 17                -1  1     90880  models.common.C3                        [256, 128, 1, False]
```

Fig 4 YOLOV5-TTA

The **YOLOv5s** backbone integrates the **ODConvBS** module, leveraging Omni-dimensional dynamic convolution with Batch Normalization and SiLU activation. This approach enhances feature extraction by employing attention mechanisms across all dimensions of the convolutional kernel space in parallel, improving flame and smoke detection accuracy [17].

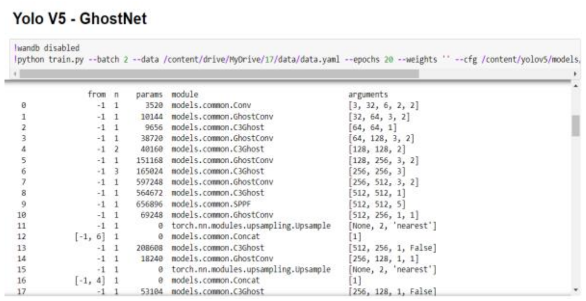


Yolo V5 - ConBS

```
!wandb disabled
!python train.py --batch 2 --data /content/drive/MyDrive/17/data/data.yaml --epochs 20 --weights '' --cfg /content/drive/MyDrive,

                 from  n    params  module                                  arguments
  0                -1  1      3520  models.common.Conv                      [3, 32, 6, 2, 2]
  1                -1  1     18560  models.common.Conv                      [32, 64, 3, 2]
  2                -1  1     18816  models.common.C3                        [64, 64, 1]
  3                -1  1     73984  models.common.Conv                      [64, 128, 3, 2]
  4                -1  2    115712  models.common.C3                        [128, 128, 2]
  5                -1  1    295424  models.common.Conv                      [128, 256, 3, 2]
  6                -1  3    625152  models.common.C3                        [256, 256, 3]
  7                -1  1   1180672  models.common.Conv                      [256, 512, 3, 2]
  8                -1  1   1182720  models.common.C3                        [512, 512, 1]
  9                -1  1    656896  models.common.SPPF                      [512, 512, 5]
 10                -1  1    131584  models.common.Conv                      [512, 256, 1, 1]
 11                -1  1         0  torch.nn.modules.upsampling.Upsample    [None, 2, 'nearest']
 12           [-1, 6]  1         0  models.common.Concat                    [1]
 13                -1  1    361984  models.common.C3                        [512, 256, 1, False]
 14                -1  1     33024  models.common.Conv                      [256, 128, 1, 1]
 15                -1  1         0  torch.nn.modules.upsampling.Upsample    [None, 2, 'nearest']
 16           [-1, 4]  1         0  models.common.Concat                    [1]
 17                -1  1     90880  models.common.C3                        [256, 128, 1, False]
```

Fig 6 YOLOV% convbs

YOLOv5 with GhostNet is a modified version of the YOLOv5 object detection model, incorporating GhostNet modules. GhostNet aims to reduce computational cost by providing a lightweight architecture with fewer parameters and computations while maintaining high accuracy. In the project, YOLOv5 GhostNet is used to strike a balance between efficiency and performance, making it suitable for real-time applications and scenarios with limited hardware resources. The integration ensures effective object detection with optimized computational efficiency.

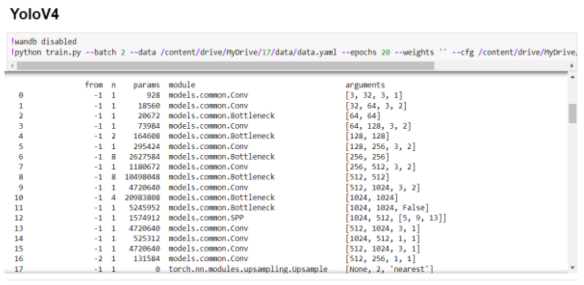

Yolo V5 - GhostNet

```
!wandb disabled
!python train.py --batch 2 --data /content/drive/MyDrive/17/data/data.yaml --epochs 20 --weights '' --cfg /content/yolov5/models.
```

```
                 from  n    params  module                                  arguments
  0                -1  1      3520  models.common.Conv                      [3, 32, 6, 2, 2]
  1                -1  1     10144  models.common.GhostConv                 [32, 64, 3, 2]
  2                -1  1      9656  models.common.C3Ghost                   [64, 64, 1]
  3                -1  1     38720  models.common.GhostConv                 [64, 128, 3, 2]
  4                -1  2     40160  models.common.C3Ghost                   [128, 128, 2]
  5                -1  1    151168  models.common.GhostConv                 [128, 256, 3, 2]
  6                -1  3    165024  models.common.C3Ghost                   [256, 256, 3]
  7                -1  1    597248  models.common.GhostConv                 [256, 512, 3, 2]
  8                -1  1    564672  models.common.C3Ghost                   [512, 512, 1]
  9                -1  1    656896  models.common.SPPF                      [512, 512, 5]
 10                -1  1     69248  models.common.GhostConv                 [512, 256, 1, 1]
 11                -1  1         0  torch.nn.modules.upsampling.Upsample    [None, 2, 'nearest']
 12           [-1, 6]  1         0  models.common.Concat                    [1]
 13                -1  1    208608  models.common.C3Ghost                   [512, 256, 1, False]
 14                -1  1     18240  models.common.GhostConv                 [256, 128, 1, 1]
 15                -1  1         0  torch.nn.modules.upsampling.Upsample    [None, 2, 'nearest']
 16           [-1, 4]  1         0  models.common.Concat                    [1]
 17                -1  1     53104  models.common.C3Ghost                   [256, 128, 1, False]
```

Fig 7 YOLOv5 with GhostNet

YOLOv4 is an advanced version of the YOLO object detection algorithm, known for its improved accuracy and efficiency. It introduces architectural enhancements, including the CSPDarknet53 backbone and PANet for feature aggregation, contributing to better object detection performance. YOLOv4 is likely used in the project for its state-of-the-art capabilities, offering high precision in flame and smoke detection [14].

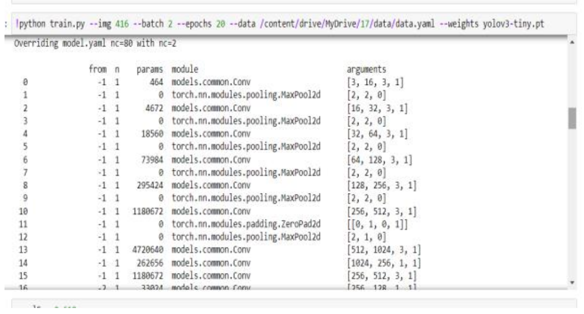

YoloV4

```
!wandb disabled
!python train.py --batch 2 --data /content/drive/MyDrive/17/data/data.yaml --epochs 20 --weights '' --cfg /content/drive/MyDrive,
```

```
                 from  n    params  module                                  arguments
  0                -1  1       928  models.common.Conv                      [3, 32, 3, 1]
  1                -1  1     18560  models.common.Conv                      [32, 64, 3, 2]
  2                -1  1     20672  models.common.Bottleneck                [64, 64]
  3                -1  1     73984  models.common.Conv                      [64, 128, 3, 2]
  4                -1  2    164608  models.common.Bottleneck                [128, 128]
  5                -1  1    295424  models.common.Conv                      [128, 256, 3, 2]
  6                -1  8   2627584  models.common.Bottleneck                [256, 256]
  7                -1  1   1180672  models.common.Conv                      [256, 512, 3, 2]
  8                -1  8  10498048  models.common.Bottleneck                [512, 512]
  9                -1  1   4720640  models.common.Conv                      [512, 1024, 3, 2]
 10                -1  4  20983808  models.common.Bottleneck                [1024, 1024]
 11                -1  1   5245952  models.common.Bottleneck                [1024, 1024, False]
 12                -1  1   1574912  models.common.SPP                       [1024, 512, [5, 9, 13]]
 13                -1  1   4720640  models.common.Conv                      [512, 1024, 3, 1]
 14                -1  1    525312  models.common.Conv                      [1024, 512, 1, 1]
 15                -1  1   4720640  models.common.Conv                      [512, 1024, 3, 1]
 16                -2  1    131584  models.common.Conv                      [512, 256, 1, 1]
 17                -1  1         0  torch.nn.modules.upsampling.Upsample    [None, 2, 'nearest']
```

Fig 8 YOLOV4

YOLOv3 is an earlier version of the YOLO series, known for its speed and accuracy in real-time object detection. It employs a Darknet-53 backbone and introduces feature pyramid networks (FPN) for better feature representation at different scales. YOLOv3 is likely included in the project for its established track record in rapid and accurate object detection, making it suitable for real-time flame and smoke detection scenarios.


```
!python train.py --img 416 --batch 2 --epochs 20 --data /content/drive/MyDrive/17/data/data.yaml --weights yolov3-tiny.pt
```

```
Overriding model.yaml nc=80 with nc=2

                 from  n    params  module                                  arguments
  0                -1  1       464  models.common.Conv                      [3, 16, 3, 1]
  1                -1  1         0  torch.nn.modules.pooling.MaxPool2d      [2, 2, 0]
  2                -1  1      4672  models.common.Conv                      [16, 32, 3, 1]
  3                -1  1         0  torch.nn.modules.pooling.MaxPool2d      [2, 2, 0]
  4                -1  1     18560  models.common.Conv                      [32, 64, 3, 1]
  5                -1  1         0  torch.nn.modules.pooling.MaxPool2d      [2, 2, 0]
  6                -1  1     73984  models.common.Conv                      [64, 128, 3, 1]
  7                -1  1         0  torch.nn.modules.pooling.MaxPool2d      [2, 2, 0]
  8                -1  1    295424  models.common.Conv                      [128, 256, 3, 1]
  9                -1  1         0  torch.nn.modules.pooling.MaxPool2d      [2, 2, 0]
 10                -1  1   1180672  models.common.Conv                      [256, 512, 3, 1]
 11                -1  1         0  torch.nn.modules.padding.ZeroPad2d      [[0, 1, 0, 1]]
 12                -1  1         0  torch.nn.modules.pooling.MaxPool2d      [2, 1, 0]
 13                -1  1   4720640  models.common.Conv                      [512, 1024, 3, 1]
 14                -1  1    262656  models.common.Conv                      [1024, 256, 1, 1]
 15                -1  1   1180672  models.common.Conv                      [256, 512, 3, 1]
 16                -2  1     33024  models.common.Conv                      [256, 128, 1, 1]
```

Fig 9 YOLOV3

SSD is a rapid object detection algorithm that predicts object locations and class probabilities in a single pass. Its efficiency makes it ideal for real-time applications, contributing to swift and efficient flame and smoke detection in the project [11].

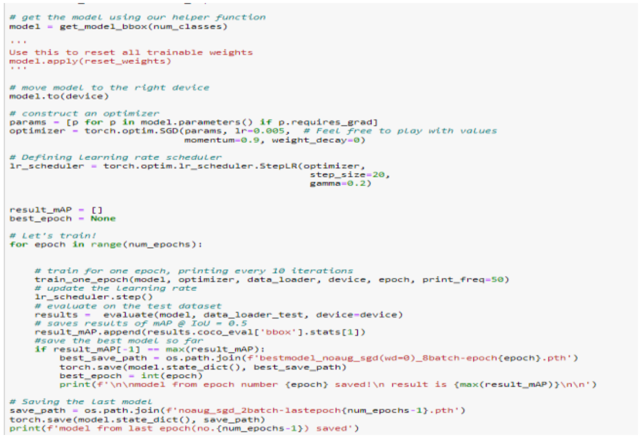
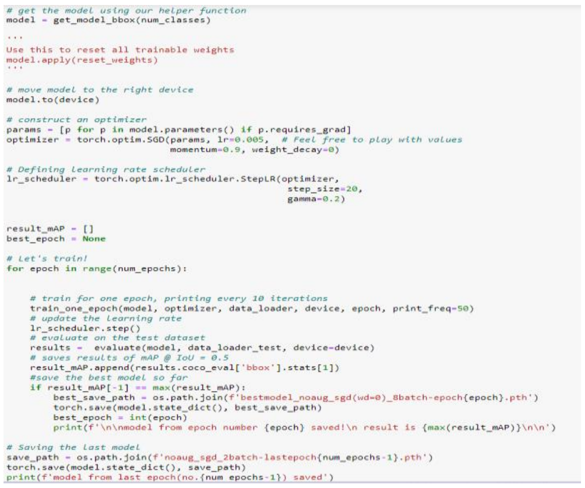

```python
# get the model using our helper function
model = get_model_bbox(num_classes)

'''
Use this to reset all trainable weights
model.apply(reset_weights)
'''

# move model to the right device
model.to(device)

# construct an optimizer
params = [p for p in model.parameters() if p.requires_grad]
optimizer = torch.optim.SGD(params, lr=0.005,  # Feel free to play with values
                            momentum=0.9, weight_decay=0)

# Defining learning rate scheduler
lr_scheduler = torch.optim.lr_scheduler.StepLR(optimizer,
                                               step_size=20,
                                               gamma=0.2)


result_mAP = []
best_epoch = None

# Let's train!
for epoch in range(num_epochs):


    # train for one epoch, printing every 10 iterations
    train_one_epoch(model, optimizer, data_loader, device, epoch, print_freq=50)
    # update the learning rate
    lr_scheduler.step()
    # evaluate on the test dataset
    results =  evaluate(model, data_loader_test, device=device)
    # saves results of mAP @ IoU = 0.5
    result_mAP.append(results.coco_eval['bbox'].stats[1])
    #save the best model so far
    if result_mAP[-1] == max(result_mAP):
        best_save_path = os.path.join(f'bestmodel_noaug_sgd(wd=0)_8batch-epoch{epoch}.pth')
        torch.save(model.state_dict(), best_save_path)
        best_epoch = int(epoch)
        print(f'\n\nmodel from epoch number {epoch} saved!\n result is {max(result_mAP)}\n\n')

# Saving the last model
save_path = os.path.join(f'noaug_sgd_2batch-lastepoch{num_epochs-1}.pth')
torch.save(model.state_dict(), save_path)
print(f'model from last epoch(no.{num_epochs-1}) saved')
```

Fig 10 SSD

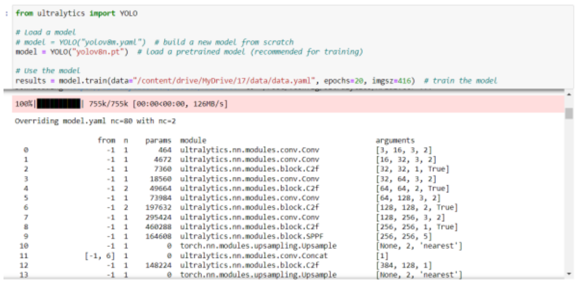
Faster R-CNN is a two-stage object detection algorithm known for its high accuracy. It proposes candidate regions in an image and refines them, offering robust detection performance. In the project, Faster R-CNN is chosen for its precision, ensuring detailed and accurate analysis of flame and smoke instances across diverse scenarios [9].



```
# get the model using our helper function
model = get_model_bbox(num_classes)

'''
Use this to reset all trainable weights
model.apply(reset_weights)
'''

# move model to the right device
model.to(device)

# construct an optimizer
params = [p for p in model.parameters() if p.requires_grad]
optimizer = torch.optim.SGD(params, lr=0.005,  # Feel free to play with values
                            momentum=0.9, weight_decay=0)

# Defining learning rate scheduler
lr_scheduler = torch.optim.lr_scheduler.StepLR(optimizer,
                                               step_size=20,
                                               gamma=0.2)


result_mAP = []
best_epoch = None

# Let's train!
for epoch in range(num_epochs):


    # train for one epoch, printing every 10 iterations
    train_one_epoch(model, optimizer, data_loader, device, epoch, print_freq=50)
    # update the learning rate
    lr_scheduler.step()
    # evaluate on the test dataset
    results =  evaluate(model, data_loader_test, device=device)
    # saves results of mAP @ IoU = 0.5
    result_mAP.append(results.coco_eval['bbox'].stats[1])
    #save the best model so far
    if result_mAP[-1] == max(result_mAP):
        best_save_path = os.path.join(f'bestmodel_noaug_sgd(wd=0)_8batch-epoch{epoch}.pth')
        torch.save(model.state_dict(), best_save_path)
        best_epoch = int(epoch)
        print(f'\n\nmodel from epoch number {epoch} saved!\n result is {max(result_mAP)}\n\n')

# Saving the last model
save_path = os.path.join(f'noaug_sgd_2batch-lastepoch{num_epochs-1}.pth')
torch.save(model.state_dict(), save_path)
print(f'model from last epoch(no.{num epochs-1}) saved')
```

Fig 11 Faster RCNN

YOLOv8 Renowned for excellence in object detection, segmentation, and pose estimation, YOLOv8 is chosen for its optimal balance of speed, accuracy, and versatility. This cutting-edge model is ideal for real-time flame and smoke detection across diverse scenarios due to its state-of-the-art performance and user-friendly design.

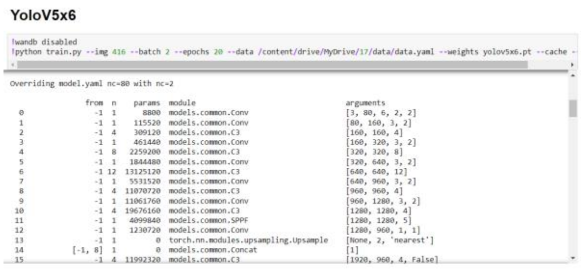


```
from ultralytics import YOLO

# Load a model
# model = YOLO("yolov8m.yaml")  # build a new model from scratch
model = YOLO("yolov8n.pt")  # load a pretrained model (recommended for training)

# Use the model
results = model.train(data="/content/drive/MyDrive/17/data/data.yaml", epochs=20, imgsz=416)  # train the model
```

```
100%|██████████| 755k/755k [00:00<00:00, 126MB/s]
Overriding model.yaml nc=80 with nc=2

                   from  n    params  module                                       arguments
  0                  -1  1       464  ultralytics.nn.modules.conv.Conv             [3, 16, 3, 2]
  1                  -1  1      4672  ultralytics.nn.modules.conv.Conv             [16, 32, 3, 2]
  2                  -1  1      7360  ultralytics.nn.modules.block.C2f             [32, 32, 1, True]
  3                  -1  1     18560  ultralytics.nn.modules.conv.Conv             [32, 64, 3, 2]
  4                  -1  2     49664  ultralytics.nn.modules.block.C2f             [64, 64, 2, True]
  5                  -1  1     73984  ultralytics.nn.modules.conv.Conv             [64, 128, 3, 2]
  6                  -1  2    197632  ultralytics.nn.modules.block.C2f             [128, 128, 2, True]
  7                  -1  1    295424  ultralytics.nn.modules.conv.Conv             [128, 256, 3, 2]
  8                  -1  1    460288  ultralytics.nn.modules.block.C2f             [256, 256, 1, True]
  9                  -1  1    164608  ultralytics.nn.modules.block.SPPF            [256, 256, 5]
 10                  -1  1         0  torch.nn.modules.upsampling.Upsample         [None, 2, 'nearest']
 11             [-1, 6]  1         0  ultralytics.nn.modules.conv.Concat           [1]
 12                  -1  1    148224  ultralytics.nn.modules.block.C2f             [384, 128, 1]
 13                  -1  1         0  torch.nn.modules.upsampling.Upsample         [None, 2, 'nearest']
```

Fig 12 YOLOv8

YOLOv5x6, a YOLOv5 variant, excels in real-time object detection due to its speed, precision, and lightweight design. Comprising CSPDarknet for feature extraction and PANet for feature fusion, it stands out for efficient and accurate flame and smoke detection, particularly deployable on mobile devices.



YoloV5x6

```
!wandb disabled
!python train.py --img 416 --batch 2 --epochs 20 --data /content/drive/MyDrive/17/data/data.yaml --weights yolov5x6.pt --cache -
```

```
Overriding model.yaml nc=80 with nc=2

                 from  n    params  module                                  arguments
  0                -1  1      8800  models.common.Conv                      [3, 80, 6, 2, 2]
  1                -1  1    115520  models.common.Conv                      [80, 160, 3, 2]
  2                -1  4    309120  models.common.C3                        [160, 160, 4]
  3                -1  1    461440  models.common.Conv                      [160, 320, 3, 2]
  4                -1  8   2259200  models.common.C3                        [320, 320, 8]
  5                -1  1   1844480  models.common.Conv                      [320, 640, 3, 2]
  6                -1 12  13125120  models.common.C3                        [640, 640, 12]
  7                -1  1   5531520  models.common.Conv                      [640, 960, 3, 2]
  8                -1  4  11070720  models.common.C3                        [960, 960, 4]
  9                -1  1  11061760  models.common.Conv                      [960, 1280, 3, 2]
 10                -1  4  19676160  models.common.C3                        [1280, 1280, 4]
 11                -1  1   4099840  models.common.SPPF                      [1280, 1280, 5]
 12                -1  1   1230720  models.common.Conv                      [1280, 960, 1, 1]
 13                -1  1         0  torch.nn.modules.upsampling.Upsample    [None, 2, 'nearest']
 14           [-1, 8]  1         0  models.common.Concat                    [1]
 15                -1  4  11992320  models.common.C3                        [1920, 960, 4, False]
```

Fig 13 YOLOV5X6

4. EXPERIMENTAL RESULTS

Precision: Precision evaluates the fraction of correctly classified instances or samples among the ones classified as positives. Thus, the formula to calculate the precision is given by:

Precision = True positives/ (True positives + False positives) = TP/(TP + FP)

$$\text{Precision} = \frac{True\ Positive}{True\ Positive + False\ Positive}$$

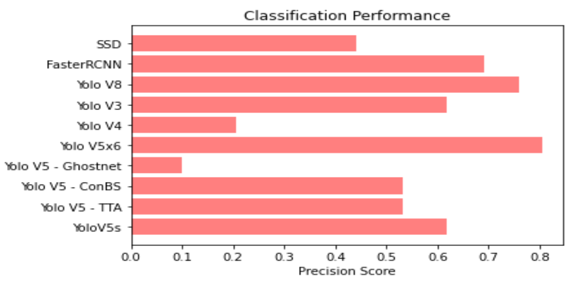


Fig 14 Precision comparison graph

Recall: Recall is a metric in machine learning that measures the ability of a model to identify all relevant instances of a particular class. It is the ratio of correctly predicted positive observations to the total actual positives, providing insights into a model's completeness in capturing instances of a given class.

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

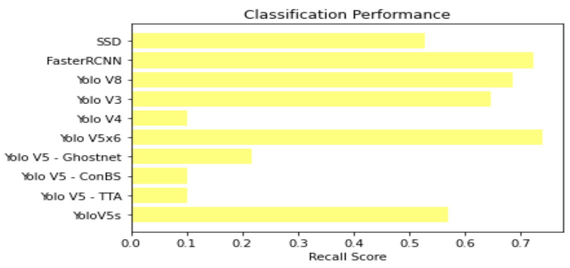


Fig 15 Recall comparison graph

mAP: Mean Average Precision (MAP) is a ranking quality metric. It considers the number of relevant recommendations and their position in the list. MAP at K is calculated as an arithmetic mean of the Average Precision (AP) at K across all users or queries.

$$mAP = \frac{1}{n}\sum_{k=1}^{k=n} AP_k$$

$$AP_k = \text{ the AP of class } k$$

$$n = \text{ the number of classes}$$

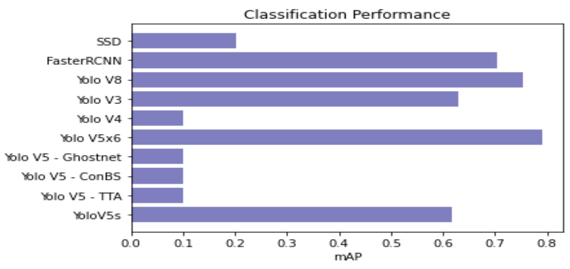
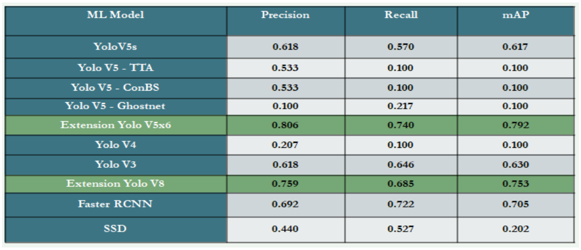
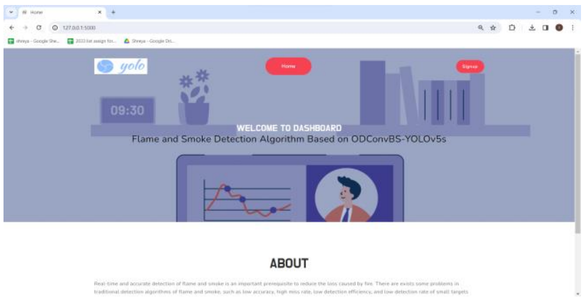


Fig 16mAP comparison graph



ML Model	Precision	Recall	mAP
YoloV5s	0.618	0.570	0.617
Yolo V5 - TTA	0.533	0.100	0.100
Yolo V5 - ConBS	0.533	0.100	0.100
Yolo V5 - Ghostnet	0.100	0.217	0.100
Extension Yolo V5x6	0.806	0.740	0.792
Yolo V4	0.207	0.100	0.100
Yolo V3	0.618	0.646	0.630
Extension Yolo V8	0.759	0.685	0.753
Faster RCNN	0.692	0.722	0.705
SSD	0.440	0.527	0.202

Fig 17 Performance Evaluation table



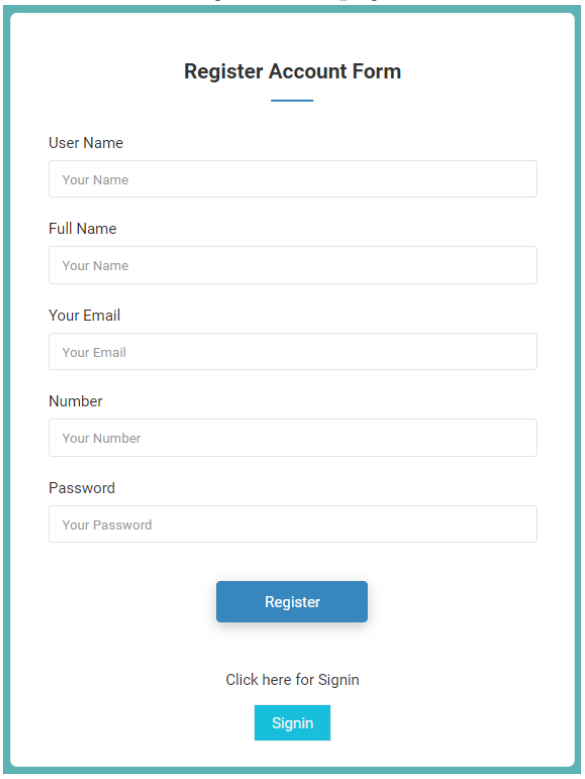
WELCOME TO DASHBOARD

Flame and Smoke Detection Algorithm Based on ODConvBS-YOLOv5s

ABOUT

Real-time and accurate detection of flame and smoke is an important prerequisite to reduce the loss caused by fire. There are exists some problems in traditional detection algorithms of flame and smoke, such as low accuracy, high miss rate, low detection efficiency, and low detection rate of small targets

Fig 18 Home page



Register Account Form

User Name

Your Name

Full Name

Your Name

Your Email

Your Email

Number

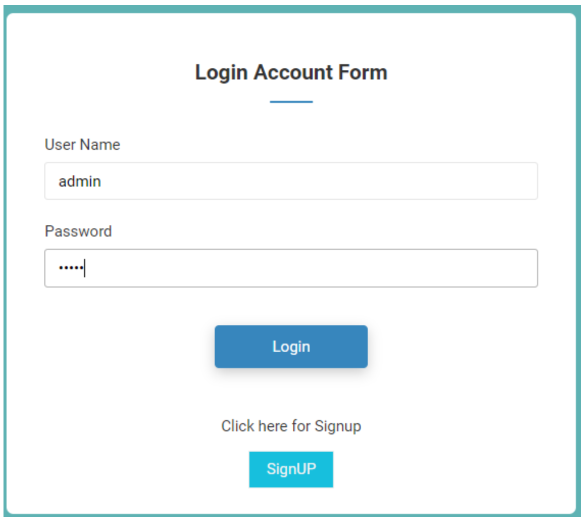
Your Number

Password

Your Password

Click here for Signin

Fig 19 Registration page



Login Account Form

User Name

admin

Password

•••••

Click here for Signup

Fig 20 Login page

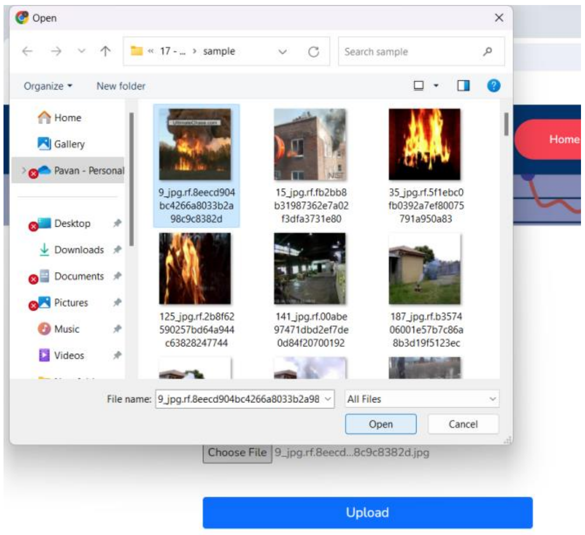

Open
17 - ... › sample
Search sample
Organize
New folder
Home
Gallery
Pavan - Personal
Desktop
Downloads
Documents
Pictures
Music
Videos
9_jpg.rf.8eecd904bc4266a8033b2a98c9c8382d
15_jpg.rf.fb2bb8b31987362e7a02f3dfa3731e80
35_jpg.rf.5f1ebc0fb0392a7ef80075791a950a83
125_jpg.rf.2b8f62590257bd64a944c63828247744
141_jpg.rf.00abe97471dbd2ef7de0d84f20700192
187_jpg.rf.b357406001e57b7c86a8b3d19f5123ec
File name: 9_jpg.rf.8eecd904bc4266a8033b2a98
All Files
Open
Cancel
Choose File 9_jpg.rf.8eecd...8c9c8382d.jpg
Upload
Home

Fig 21 Input image folder

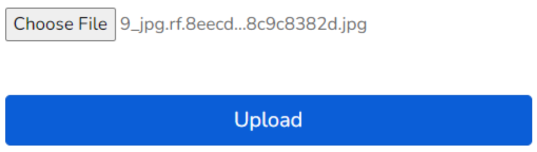


Fig 22 Upload input image

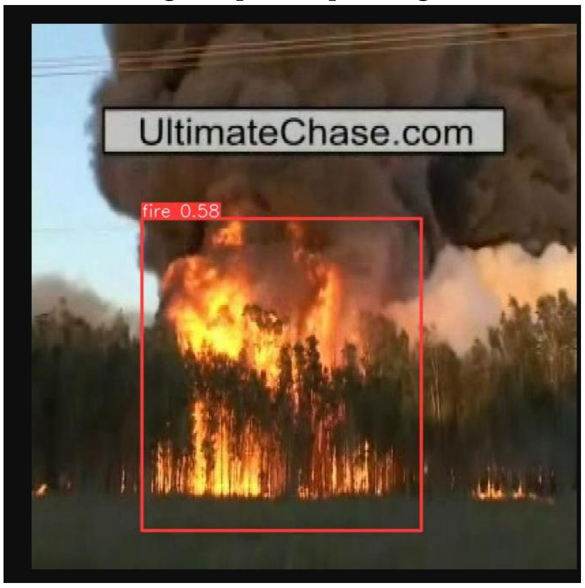


Fig 23 Predict result for given input

5. CONCLUSION

Through the exploration of various state-of-the-art models, including YOLO V5 variants, YOLO V8, SSD, and FasterRCNN [9], [10], [11], [12]the project achieved a comprehensive understanding of their performance in detecting flames and smoke in remote aerial satellite images. The project extended its scope by evaluating the ODConvBS-YOLOv5s model, specifically designed for flame and smoke detection. This specialized algorithm was compared with other models to assess its effectiveness in improving detection accuracy. The implementation of a Flask-based web interface with SQLite authentication enhances user experience, making it accessible for individuals seeking an intuitive platform for satellite image analysis. YOLO v5x6 which is an extension , boasting an impressive mAP of 0.792, proves to be an exceptional solution for real-time flame and smoke detection. Its superior performance establishes it as a leading algorithm in object detection, demonstrating reliability and efficiency for diverse applications. The developed system not only serves the specific purpose of flame and smoke detection in remote aerial satellite images but also lays the foundation for future extensions and applications in satellite image analysis, benefiting both researchers and end-users in diverse fields.

6. FUTURE SCOPE

Explore the incorporation of additional sensor data, such as infrared and thermal imaging, to enhance the system's ability to detect flame and smoke in various environmental conditions, including low-light or obscured scenarios. Extend the project to include a real-time decision support system that utilizes the detected flame and smoke data to provide timely information for emergency response teams, facilitating quicker and more informed decision-making during fire incidents. Investigate the deployment of the detection algorithm on edge computing devices to promote decentralized processing. This can improve the system's responsiveness and reduce dependence on centralized servers, making it more suitable for distributed applications. Regularly update and expand the dataset with diverse flame and smoke scenarios to continuously train and improve the model's performance [3,4,5]. This ensures adaptability to evolving environmental conditions and emerging challenges in fire detection, contributing to a robust and future-ready system.

REFERENCES

[1] Z.-Q. Zhao, P. Zheng, S.-T. Xu, and X. Wu, ''Object detection with deep learning: A review,'' IEEE Trans. Pattern Anal. Mach. Intell., vol. 30, no. 11, pp. 3212–3232, Nov. 2019.

[2] X. Wu, D. Sahoo, and S. C. Hoi, ''Recent advances in deep learning for object detection,'' Neurocomputing, vol. 396, pp. 39–64, Jul. 2020.

[3] J. Chen, Y. He, and J. Wang, ''Multi-feature fusion based fast video flame detection,'' Building Environ., vol. 45, no. 5, pp. 1113–1122, May 2010.

[4] A. Pawar, ''A multi-disciplinary vision-based fire and smoke detection system,'' in Proc. 4th Int. Conf. Electron., Commun. Aerosp. Technol. (ICECA), Nov. 2020, pp. 900–904.

[5] S. Khan, K. Muhammad, and T. Hussain, ''Deepsmoke: Deep learning model for smoke detection and segmentation in outdoor environments,'' Expert Syst. Appl., vol. 182, Nov. 2021, Art. no. 115125.

[6] G. Lin, Y. Zhang, G. Xu, and Q. Zhang, ''Smoke detection on video sequences using 3D convolutional neural networks,'' Fire Technol., vol. 55, no. 5, pp. 1827–1847, Sep. 2019.

[7] L. Wang, W. Li, and W. Li, ''Appearance-and-relation networks for video classification,'' in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., Jun. 2018, pp. 1430–1439.

[8] P. Li and W. Zhao, ''Image fire detection algorithms based on convolutional neural networks,'' Case Stud. Thermal Eng., vol. 19, Jun. 2020, Art. no. 100625.

[9] S. Ren, K. He, and R. Girshick, ''Towards real-time object detection with region proposal networks,'' 2019, arXiv:1506.01497.

[10] J. Dai, Y. Li, and K. He, ''R-FCN: Object detection via region-based fully convolutional networks,'' in Proc. Adv. Neural Inf. Process. Syst., vol. 29, 2016, pp. 1–11.

[11] W. Liu, D. Anguelov, and D. Erhan, ''SSD: Single shot multibox detector,'' in Proc. Eur. Conf. Comput. Vis. Cham, Switzerland: Springer, 2016, pp. 21–37.

[12] J. Redmon and A. Farhadi, ''YOLOv3: An incremental improvement,'' 2018, arXiv:1804.02767.

[13] S. Saponara, A. Elhanashi, and A. Gagliardi, ''Real-time video fire/smoke detection based on CNN in antifire surveillance systems,'' J. Real-Time Image Process., vol. 18, no. 3, pp. 889–900, 2021.

[14] M. Torabian, H. Pourghassem, and H. Mahdavi-Nasab, ''Fire detection based on fractal analysis and spatio-temporal features,'' Fire Technol., vol. 57, pp. 2583–2614, May 2021.

[15] K. Avazov, M. Mukhiddinov, F. Makhmudov, and Y. I. Cho, ''Fire detection method in smart city environments using a deep-learning-based approach,'' Electronics, vol. 11, p. 73, Dec. 2022.

[16] S. Majid, F. Alenezi, S. Masood, M. Ahmad, E. S. Gündüz, and K. Polat, ''Attention based CNN model for fire detection and localization in real-world images,'' Expert Syst. Appl., vol. 189, Mar. 2022, Art. no. 116114.

[17] Z. Xue, H. Lin, and F. Wang, ''A small target forest fire detection model based on YOLOv5 improvement,'' Forests, vol. 13, no. 8, p. 1332, Aug. 2022.

[18] Y. Hu, J. Zhan, and G. Zhou, ''Fast forest fire smoke detection using MVMNet,'' Knowl.-Based Syst., vol. 241, Jan. 2022, Art. no. 108219.

[19] J. Li, G. Zhou, and A. Chen, ‘‘Adaptive linear feature-reuse network for rapid forest fire smoke detection model,’’ Ecolog. Informat., vol. 68, May 2022, Art. no. 101584.
[20] O. Khudayberdiev, J. Zhang, and S. M. Abdullahi, ‘‘Light-FireNet: An efficient lightweight network for fire detection in diverse environments,’’ Multimedia Tools Appl., vol. 81, pp. 24553–24572, Mar. 2022.
[21] A. Hosseini, M. Hashemzadeh, and N. Farajzadeh, ‘‘UFS-Net: A unified flame and smoke detection method for early detection of fire in video surveillance applications using CNNs,’’ J. Comput. Sci., vol. 61, May 2022, Art. no. 101638.
[22] T. Liang, H. Bao, and W. Pan, ‘‘Traffic sign detection via improved sparse R-CNN for autonomous vehicles,’’ J. Adv. Transp., vol. 2022, pp. 1–16, Mar. 2022.
[23] Y. Gu and B. Si, ‘‘A novel lightweight real-time traffic sign detection integration framework based on YOLOv4,’’ Entropy, vol. 24, no. 4, p. 487, Mar. 2022.
[24] J. Wang, Y. Chen, and Z. Dong, ‘‘Improved YOLOv5 network for realtime multi-scale traffic sign detection,’’ Neural Comput. Appl., vol. 35, pp. 7853–7865, Dec. 2022.
[25] F. Dadboud, V. Patel, and V. Mehta, ‘‘Single-stage UAV detection and classification with YOLOV5: Mosaic data augmentation and PANet,’’ in Proc. 17th IEEE Int. Conf. Adv. Video Signal Based Surveill. (AVSS), Nov. 2021, pp. 1–8.
[26] C. Si, Z. Zhang, and F. Qi, ‘‘Better robustness by more coverage: Adversarial and mixup data augmentation for robust finetuning,’’ in Proc. Findings Assoc. Comput. Linguistics, (ACL-IJCNLP), 2021, pp. 1569–1576.
[27] Z. Gevorgyan, ‘‘SIoU loss: More powerful learning for bounding box regression,’’ 2022, arXiv:2205.12740.
[28] T. Y. Lin, P. Dollár, and R. Girshick, ‘‘Feature pyramid networks for object detection,’’ in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., Jul. 2017, pp. 2117–2125.
[29] Z. Zheng, P. Wang, and D. Ren, ‘‘Enhancing geometric factors in model learning and inference for object detection and instance segmentation,’’ IEEE Trans. Cybern., vol. 52, no. 8, pp. 8574–8586, Aug. 2022.
[30] C. Li, A. Zhou, and A. Yao, ‘‘Omni-dimensional dynamic convolution,’’ 2022, arXiv:2209.07947.
[31] Q. L. Zhang and Y. B. Yang, ‘‘SA-Net: Shuffle attention for deep convolutional neural networks,’’ in Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP), Jun. 2021, pp. 2235–2239.
[32] N. Ma, X. Zhang, and H. T. Zheng, ‘‘ShuffleNet V2: Practical guidelines for efficient CNN architecture design,’’ in Proc. Eur. Conf. Comput. Vis. (ECCV), Sep. 2018, pp. 116–131.
[33] Y. Rao, W. Zhao, Y. Tang, J. Zhou, S.-N. Lim, and J. Lu, ‘‘HorNet: Efficient high-order spatial interactions with recursive gated convolutions,’’ 2022, arXiv:2207.14284.
[34] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby, ‘‘An image is worth 16×16 words: Transformers for image recognition at scale,’’ 2020, arXiv:2010.11929.