REAL-TIME FIRE DETECTION

A PROJECT REPORT

Submitted by

SAKTHI SREE V [REGISTER NO:211414104229]
SOFIYA S [REGISTER NO:211414104260]
SWATHI P[REGISTER NO:211419104281]

in partial fulfillment for the award of the degree

of

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE AND ENGINEERING

PANIMALAR ENGINEERING COLLEGE, CHENNAI-600123.

ANNA UNIVERSITY: CHENNAI 600 025

APRIL 2023

PANIMALAR ENNGINEERING COLLEGE
(An Autonomous Institution, Affiliated to Anna University, Chennai)

BONAFIDE CERTIFICATE

Certified that this project report **"REAL-TIME FIRE DETECTION"** is the bonafide work of "**SAKTHI SREE V (211419104229), SOFIYA S (211419104260) AND SWATHI P (211419104281)"** who carried out the project work under my supervision.

SIGNATURE

Dr. L. JABASHEELA, M.E., Ph.D.,
HEAD OF THE DEPARTMENT

DEPARTMENT OF CSE,
PANIMALAR ENGINEERING COLLEGE,
NASARATHPETTAI,
POONAMALLEE,
CHENNAI-600 123.

SIGNATURE

Dr. A. HEMLATHADHEVI, M.E., Ph.D.,
ASSOCIATE PROFESSOR

DEPARTMENT OF CSE,
PANIMALAR ENGINEERING COLLEGE,
NASARATHPETTAI,
POONAMALLEE,
CHENNAI-600 123.

Certified that the above candidate(s) was/ were examined in the Anna University Project Viva-Voce Examination held on 11.04.2023

INTERNAL EXAMINER **EXTERNAL EXAMINER**

DECLARATION BY THE STUDENT

We hereby declare that this project report titled **"SAKTHI SREE V (211419104229), SOFIYA S (211419104260) AND SWATHI P (211419104281)"**, under the guidance of **Dr. A. HEMLATHADHEVI** is the original work done by us and we have not plagiarized or submitted to any other degree in any university by us.

SAKTHI SREE V
SOFIYA S
SWATHI P

ACKNOWLEDGEMENT

We would like to express our deep gratitude to our respected Secretary and Correspondent **Dr. P. CHINNADURAI, M.A., Ph.D.,** for his kind words and enthusiastic motivation, which inspired us a lot in completing this project.

We express our sincere thanks to our beloved Directors
Tmt. C. VIJAYARAJESWARI, **Dr. C. SAKTHI KUMAR, M.E., Ph.D**., and
Dr. SARANYASREE SAKTHI KUMAR B.E., M.B.A., Ph.D., for providing us with the necessary facilities to undertake this project.

We also express our gratitude to our Principal **Dr. K. Mani, M.E., Ph.D.,** who facilitated us in completing the project.

We thank the Head of the CSE Department, **Dr. L. JABASHEELA, M.E., Ph.D.,** for the support extended throughout the project.

We would like to thank my **Dr. A. HEMLATHADHEVI, M.E., Ph.D.,** and all the faculty members of the Department of CSE for their advice and encouragement for the successful completion of the project.

SAKTHI SREE V
SOFIYA S
SWATHI P

ABSTRACT

Fire, an hazardous incident, poses a significant risk as it can be destructive and dangerous, causing wildfires, building fires, and explosions. A fire can burn out of control, engulfing the area in heat and toxic with heavy smoke. So, it's crucial to spot and put out fire at its very beginning. The majority of fire monitoring and alarm systems now in use rely on sensors, which have drawbacks such as low sensitivity, sluggish response, limited coverage, and poor stability. This explores a way to develop a fire localization system using yolo5 network. As security technology has advanced, cctv cameras are now mounted for surveillance at the majority of buildings and traffic lights. Early fire detection and identification are possible using the camera's live footage. The main aim is to propose a fire detection system based on the YOLO (You Only Look Once) object detection algorithm and OpenCV (Open Source Computer Vision Library). The system takes input from a video stream and uses YOLO to identify objects in the image, including fires. OpenCV is then used to track the identified objects and provide continuous fire detection. The proposed system is evaluated on a dataset of videos containing various types of fires, and the results demonstrate that it achieves high accuracy and real-time performance. The system can be used in a variety of applications, including fire detection in buildings, forests, and industrial environments. The proposed system provides a cost-effective and efficient solution for fire detection, which can help reduce the risk of fire-related accidents and save lives.YOLOV5 network is used for the successful detection and warning of fire disasters. So, the moment the fire occurs, it is detected precisely and alarmed. Thus, a quick and extremely accurate fire detection could be employed day and night, regardless of size and shape was made possible by using the yolo5 algorithm.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

ABBREVIATION	EXPANSION
YOLO	YOU ONLY LOOK ONCE
CV	COMPUTER VISION

CHAPTER 1
INTRODUCTION

CHAPTER 1
INTRODUCTION

1.1 INTRODUCTION:

People's lives are seriously at risk from fires. In order to reduce the harm they inflict, it is crucial to identify fires as soon as possible. Traditionally,fire detection has been done by human observation. It is, however, a time- and labour-intensive process.

Since technology has advanced, sensors are now used, however sensor-based detection systems are best used in tiny indoor settings. Smoke sensors, gas sensors, temperature sensors, humidity sensors, integrated sensors, and other types of sensors are utilised for forest fire detection.However, the device's detecting range is constrained, installation costs are considerable, and it also has to deal with challenging power supply and networking issues. Additionally, the sensors are unable to deliver crucial visual information that would enable firefighters to quickly assess the situation at the fire scene. As a result, this approach might not be appropriate. As hardware and software technologies advance and computer arithmetic is improved, more researchers are starting to look at using deep learning to identify forest fires in order to circumvent the limitations of sensors. YOLO is a well-known real-time object detection technique. (You Only Look Once). For small objects, it is known to have lower detection accuracy.

Deep learning and machine vision are both crucial in the detection of fires. The YOLOv5 (You Only Look Once version 5) object detection model employs deep neural networks to identify items in real-time. It is a cutting-edge algorithm that is quicker and more precise than earlier iterations. It is feasible to create an effective and dependable fire detection system by combining YOLOv5 with computer vision and deep learning methods.

1.2 PROBLEM DEFINITION:

The issue of fire detection is an important challenge for public safety since it can cause significant damage to property and human life. Although there are numerous fire detection systems in place, they often have limitations that need to be addressed. Specifically, current systems may face challenges in detecting fires in low-light conditions or detecting multiple fires simultaneously. In addition, there may be issues with false alarms, which can waste valuable resources and cause unnecessary panic. To address these challenges, the implementation of an advanced fire detection system that incorporates artificial intelligence (AI) and machine learning (ML) algorithms is necessary. Specifically, the YOLOv5 architecture can be utilized to develop a fire detection system that accurately detects fires in real-time, improves detection in low-light conditions, and reduces false alarms. This project aims to develop a robust fire detection system that can help to improve public safety and prevent the loss of life and property caused by fires.

CHAPTER 2
LITERATURE SURVEY

CHAPTER 2
LITERATURE SURVEY

TITLE 1: Forest fire and smoke detection using deep learning – based learning without forgetting.

YEAR: 2023

ABSTRACT:

Forests are an essential natural resource to humankind, providing a myriad of direct and indirect benefits. Natural disasters like forest fires have a major impact on global warming and the continued existence of life on Earth. Automatic identification of forest fires is thus an important feld to research in order to minimize disasters. Early fire detection can also help decision-makers plan mitigation methods and extinguishing tactics. This research looks at fire/smoke detection from images using AI-based computer vision techniques. Convolutional Neural Networks (CNN) are a type of Artificial Intelligence (AI) approach that have been shown to outperform state-of-the-art methods in image classification and other computer vision tasks, but their training time can be prohibitive. Further, a pretrained CNN may underperform when there is no sufficient dataset available. To address this issue, transfer learning is exercised on pre-trained models. However, the models may lose their classification abilities on the original datasets when transfer learning is applied. To solve this problem, we use learning without forgetting (LwF), which trains the network with a new task but keeps the network's pre-existing abilities intact.

TITLE 2: Fire-Net: A Deep Learning Framework for Active Forest Fire Detection

YEAR: 2022

ABSTRACT:

Forest conservation is crucial for the maintenance of a healthy and thriving ecosystem. The field of remote sensing (RS) has been integral with the wide adoption of computer vision and sensor technologies for forest land observation. One critical area of interest
is the detection of active forest fires. A forest fire, which occurs naturally or manually induced, can quickly sweep through vast amounts of land, leaving behind unfathomable damage and loss of lives. Automatic detection of active forest fires (and burning
biomass) is hence an important area to pursue to avoid unwanted catastrophes. Early fire detection can also be useful for decision makers to plan mitigation strategies as well as extinguishing efforts. In this paper, we present a deep learning
framework called Fire-Net, that is trained on Landsat-8 imagery for the detection of active fires and burning biomass.Specifically, we fuse the optical (Red, Green, and Blue) and thermal modalities from the images for a more effective
representation. In addition, our network leverages the residual convolution and separable convolution blocks, enabling deeper features from coarse datasets to be extracted. Experimental results show an overall accuracy of 97.35%, while also being able to
robustly detect small active fires. The imagery for this study is taken from Australian and North American forests regions, the Amazon rainforest, Central Africa and Chernobyl (Ukraine), where forest fires are actively reported.

TITLE 3: Characteristics Based Fire Detection System Under the Effect of Electric Fields With ImprovedYolo-v4 and ViBe.

YEAR: 2022

ABSTRACT:

To address slow image-based detection of fires under the effect of electric fields and its limitation to static fire characteristics, this paper proposes video-based fire detection system with improved you only look once version 4 (Yolo-v4) and visual background extractor (ViBe) algorithms. The proposed system uses a simplified weighted bi-directional feature pyramid network (Bi-FPN) in place of the path aggregation network (PANet) as a feature fusion network in Yolo-v4. Using multiple dynamic fire characteristics, it can eliminate falsely detected frames. The ViBe algorithm is improved to consider the sudden change of light triggered by fire flickering. Compared with other fire detection algorithms, the proposed system achieves 98.9% fire detection accuracy with a false detection rate of 2.2%. It can extract target fires by adjusting to sudden changes of light using no more than 16 frames. Moreover, the system achieves fire detection with more dynamic fire characteristics compared with the image-based fire detection under the effect of electric fields.

TITLE 4: Wildland Fire Detection and Monitoring Using a Drone-Collected RGB/IR Image Dataset

YEAR: 2022

ABSTRACT:

Fire is an abnormal event which can cause significant damage to lives and property. In this paper, we propose a deep learning-based fire

detection method using a video sequence, which imitates the human fire detection process. The proposed method uses Faster Region-based Convolutional Neural Network (R-CNN) to detect the suspected regions of fire (SRoFs) and of non-fire based on their spatial features. Then, the summarized features within the bounding boxes in successive frames are accumulated by Long Short-Term Memory (LSTM) to classify whether there is a fire or not in a short-term period. The decisions for successive short-term periods are then combined in the majority voting for the final decision in a long-term period. In addition, the areas of both flame and smoke are calculated and their temporal changes are reported to interpret the dynamic fire behaviour with the final fire decision. Experiments show that the proposed long-term video-based method can successfully improve the fire detection accuracy compared with the still image-based or short-term video-based method by reducing both the false detections and the misdetections.

TITLE 5: A fast video fire detection of irregular burning features in fire flame using indoor fire sensing robots.

YEAR: 2022

ABSTRACT:

This paper proposes a fully convolutional variational autoencoder (VAE) for features extraction from a large-scale dataset of fire images. The dataset will be used to train the deep learning algorithm to detect fire and smoke. The features extraction is used to tackle the curse of dimensionality, which is the common issue in training deep learning with huge datasets. Features extraction aims to reduce the dimension of the dataset significantly without losing too much essential information. Variational autoencoders (VAEs) are powerful generative model, which can be used for dimension reduction. VAEs work better than any other methods available for this purpose because they can explore variations on the data in a specific direction

TITLE 6: Fully convolutional variational auto encoder for feature extraction of fire detection system.

YEAR: 2020

ABSTRACT:

Current forest monitoring technologies including satellite remote sensing, manned/piloted aircraft, and observation towers leave uncertainties about a wildfire's extent, behavior, and conditions in the fire's near environment, particularly during its early growth. Rapid mapping and real-time fire monitoring can inform in-time intervention or management solutions to maximize beneficial fire outcomes. Drone systems' unique features of 3D mobility, low flight altitude, and fast and easy deployment make them a valuable tool for early detection and assessment of wildland fires, especially in remote forests that are not easily accessible by ground vehicles. In addition, the lack of abundant, well-annotated aerial datasets – in part due to unmanned aerial vehicles' (UAVs') flight restrictions during prescribed burns and wildfires – has limited research advances in reliable data-driven fire detection and modelling techniques. While existing wildland fire datasets often include either colour or thermal fire images, here we present (1) a multi-modal UAV-collected dataset of dual-feed side-by-side videos including both RGB and thermal images of a prescribed fire in an open canopy pine forest in Northern Arizona and (2) a deep learning-based methodology for detecting fire and smoke pixels at accuracy much higher than the usual single-channel video feeds. The collected images are labelled to ''fire'' or ''no-fire'' frames by two human experts using side-by-side RGB and thermal images to determine the label. To provide context to the main dataset's aerial imagery, the included supplementary dataset provides a georeferenced pre-burn point cloud, an RGB orthoscopic, weather information, a burn plan, and other burn information. By using

and expanding on this guide dataset, research can develop new data-driven fire detection, fire segmentation, and fire modelling techniques.

TITLE 7: A Video-Based Fire Detection Using Deep Learning Models

YEAR: 2019

ABSTRACT:

This paper proposes a novel method for fire and smoke detection using video images. The ViBe method is used to extract a background from the whole video and to update the exact motion areas using frame-by-frame differences. Dynamic and static features extraction are combined to recognize the fire and smoke areas. For static features, we use deep learning to detect most of fire and smoke areas based on a Caffemodel. Another static feature is the degree of irregularity of fire and smoke. An adaptive weighted direction algorithm is further introduced to this paper. To further reduce the false alarm rate and locate the original fire position, every frame image of video is divided into 16×16 grids and the times of smoke and fire occurrences of each part is recorded. All clues are combined to reach a final detection result. Experimental results show that the proposed method in this paper can efficiently detect fire and smoke and reduce the loss and false detection rates.

TITLE 8: Fire detection using computer vision.

YEAR: 2019

ABSTRACT:

This paper provides a computer vision based technique for detecting fire and identifying hazardous fire by processing the video data generated by an ordinary camera. While there are copious amount of publications in fire detection with images, distinguishing hazardous fire

from non-hazardous is a problem that is still unsolved. The proposed technique uses the colour and fluctuation characteristics of fire to detect it. Initially the algorithm locates regions of the video where there is motion; from these regions fire coloured pixels are extracted and then wavelet transform is applied to confirm that the moving object is fire. The proposed technique tracks the rate of increment of fire region to distinguish between hazardous and controlled fire. Experimental results demonstrate that the proposed technique is effective in detecting the fire and classifying it as hazardous or controlled. The performance of the technique was observed in terms of true positive rate which was observed to be 85.57% while the average delay in detection of hazardous fire was 66.2 frames at a frame rate of 10 frames per second or 6.62 seconds.

TITLE 9: Fire Detection in H.264 Compressed video

YEAR: 2019

ABSTRACT:

In this paper, we propose a compressed domain fire detection algorithm using macroblock types and Markov Model in H.264 video. Compressed domain method does not require decoding to pixel domain, instead a syntax parser extracts syntax elements which are only available in compressed domain. Our method extracts only macroblock type and corresponding macroblock address information. Markov model with fire and non-fire models are evaluated using offline trained data. Our experiments show that the algorithm is able to detect and identify fire event in compressed domain successfully, despite a small chunk of data is used in the process.

TITLE 10: An Adaptive Threshold Deep Learning Method for Fire and Smoke Detection

YEAR: 2017

ABSTRACT:

This paper proposes a novel method for fire and smoke detection using video images. The ViBe method is used to extract a background from the whole video and to update the exact motion areas using frame-by-frame differences. Dynamic and static features extraction are combined to recognize the fire and smoke areas. For static features, we use deep learning to detect most of fire and smoke areas based on a Caffemodel. Another static feature is the degree of irregularity of fire and smoke. An adaptive weighted direction algorithm is further introduced to this paper. To further reduce the false alarm rate and locate the original fire position, every frame image of video is divided into 16×16 grids and the times of smoke and fire occurrences of each part is recorded. All clues are combined to reach a final detection result. Experimental results show that the proposed method in this paper can efficiently detect fire and smoke and reduce the loss and false detection rates.

TITLE 11: A Small Target Forest Fire Detection Model Based on YOLOv5 Improvement

YEAR: 2022

ABSTRACT:

Very unpredictable and dangerous, forest fires. The apparent limits of manual inspection techniques, sensor-based detection, satellite remote sensing, and computer vision detection are all present. Deep learning methods may learn from data and extract characteristics in an adaptable way of wildfires. Nevertheless, the model is unable to gain useful information due to the tiny size of the forest fire target in the long-range collected forest fire photographs. We suggest an enhanced YOLOv5-based small-target identification model for forest fires to address this

issue. For practical applications, this model needs cameras as sensors for spotting forest fires. To better focus on the global data of tiny forest fire targets, we first enhanced the Backbone layer of YOLOv5 and changed the original Spatial Pyramid Pooling-Fast (SPPF) module of YOLOv5 to the Spatial Pyramid Pooling-Fast-Plus (SPPFP) module. Later, to increase the recognition of tiny forest fire targets, we included the Convolutional Block Attention Module (CBAM) attention module. Second, the Path Aggregation Network (PANet) was modified to the Bi-directional Feature Pyramid Network and a very-small-target detection layer was added to the YOLOv5 Neck layer (BiFPN). Lastly, a migration learning technique was employed for training because the original small-target forest fire dataset is a tiny sample dataset. The modified structure in this work enhances mAP@0.5 by 10.1%, according to experimental results on a first small-target forest fire dataset we developed. This shows that the performance of our suggested model has been significantly improved and has some potential for implementation.

CHAPTER 3
SYSTEM ANALYSIS

CHAPTER 3
SYSTEM ANALYSIS

3.1 EXISTING SYSTEM:

Smoke alarms and heat alarms are being used to detect fires. One module is not enough to monitor all of the potential fire prone areas, which is the fundamental drawback of smoke sensor alarms and heat sensor alarms.

The only way to avoid a fire is to exercise caution at all times. Even if they are installed in every nook and cranny, it simply is not enough to constantly produce an efficient output. As the number of smoke sensors required rises, the price will rise by a factor of multiples. Within seconds of an accident or a fire, the suggested system can generate reliable and highly accurate alert. Since, only one piece of software is required to run the complete monitoring network, costs are reduced.

DRAWBACKS:

- Expensive and struggle to detect small fire.
- It has Low accuracy, and take more time to detect fire.
- Fails to detect fire at distant.

3.2 PROPOSED SYSTEM:

The proposed model combines both open cv and deep learning algorithm(yolo5) to efficiently detect fire.Yolov5 is faster than its earlier versions and offers improved speed and accuracy for object detection, classification and localization. It is a light weight model and it is easy to deploy that makes it suitable for real time application.
Pre-trained weights and configuration files of yolo5 algorithm is loaded into the webcam for image processing and fire detection. The input obtained from webcam is resized to fit the network size of yolo5 algorithm. Then, it is fed into the yolo5 network where object detection and labelling occurs. If the labelling matches with the class fire, the system alerts with an alarm sound.

ADVANTAGES:

- Accurate detection: YOLOv5 is a state-of-the-art object detection algorithm that uses deep learning to accurately detect fire in real-time. By combining this algorithm with computer vision techniques, the system can identify and locate fires with high accuracy, reducing false alarms and improving response times.

- Fast response times: The real-time detection capabilities of YOLOv5 combined with computer vision techniques enable the system to quickly detect fires and alert emergency responders, enabling them to take prompt action and potentially save lives and property.

- Robustness to environmental conditions: Fire detection systems based on YOLOv5 and computer vision are less prone to false alarms caused by environmental conditions such as smoke, shadows, and low light levels. The deep learning algorithms can

learn to distinguish between real fires and these other factors, improving the overall reliability of the system.

- Scalability: YOLOv5 and computer vision techniques can be easily scaled up to cover large areas such as warehouses, factories, and other industrial facilities. This makes them suitable for use in a wide range of applications where traditional fire detection systems may be less effective.

- Cost-effectiveness: YOLOv5 and computer vision techniques can be implemented using off-the-shelf hardware, reducing the cost of deploying fire detection systems. Additionally, the accuracy of these systems reduces false alarms, potentially saving costs associated with unnecessary emergency responses.

3.3 FEASIBILITY STUDY:

A feasibility study for a fire detection system using computer vision (CV) would involve assessing the technical and economic viability of using CV-based algorithms and technologies to detect fires. The following are some of the key factors that would need to be considered:

Technical Feasibility:

- Identifying the most suitable CV-based fire detection technology, such as video-based fire detection, thermal imaging, or multispectral imaging.
- Assessing the reliability and accuracy of the CV-based system in detecting fires, as well as its ability to detect and differentiate between various types of fires and environmental factors that may affect the accuracy of the system.
- Evaluating the compatibility and integration of the CV-based system with other building systems, such as fire suppression, evacuation, and notification systems.

Economic Feasibility:

- Estimating the initial investment cost of installing the CV-based fire detection system, including equipment, installation, and commissioning.
- Determining the ongoing maintenance and operating costs of the system, such as periodic testing, repairs, and replacement of components.
- Analyzing the potential benefits of the system, such as reducing the risk of property damage and loss, minimizing business interruption, and enhancing occupant safety, and comparing them with the costs to determine the system's overall financial viability.

Legal and Regulatory Compliance:

- Identifying and complying with relevant local, national, and international standards and regulations governing the design, installation, and operation of fire detection systems, such as NFPA, IEC, UL, FM, and EN.
- Ensuring that the CV-based fire detection system is appropriately certified and approved by the relevant authorities.
- Environmental Considerations:
- Evaluating the environmental impact of the CV-based fire detection system, such as energy consumption, emissions, waste, and disposal of equipment and components, and taking measures to mitigate any adverse effects.
- Once these factors have been evaluated, a feasibility study report can be prepared, outlining the findings and recommendations regarding the implementation of the CV-based fire detection system. The report should also highlight any potential risks and challenges associated with the system and propose mitigation strategies to address them.

3.4 HARDWARE ENVIRONMENT

- ✓ Processor - I5
- ✓ RAM - 8 GB(min)
- ✓ Hard Disk - 500 GB
- ✓ Key Board - Standard Windows Keyboard
- ✓ Mouse - Two or Three Button Mouse
- ✓ Monitor - LCD

3.5 SOFTWARE ENVIRONMENT:

- ✓ Operating System - Linux, Windows/7/10
- ✓ Server - Anaconda, VS code
- ✓ Serve side Script - Python, AIML

3.6 SOFTWARE DESCRIPTION

3.6.1 OPEN CV

OpenCV (Open Source Computer Vision) is a popular open-source library for computer vision and machine learning algorithms. It is written in C++ and provides a rich set of functions and tools for image and video processing, object detection, face recognition, and many other applications. OpenCV supports a wide range of platforms, including Windows, Linux, macOS, Android, and iOS.

OpenCV provides a collection of powerful functions and algorithms for image and video processing, such as filtering, thresholding, morphological operations, edge detection, and feature extraction. It also includes popular machine learning algorithms for object detection and recognition, such as Haar cascades and deep neural

networks. These algorithms can be used for a variety of applications, including security, surveillance, robotics, and medical imaging.

OpenCV is widely used in academia and industry for research and development purposes. It is used in many real-world applications, such as autonomous vehicles, augmented reality, and video surveillance systems. The library has a large community of developers and users who contribute to its development and maintenance.

OpenCV can be used in a variety of programming languages, including C++, Python, Java, and MATLAB. It also provides interfaces for popular deep learning frameworks, such as TensorFlow and PyTorch, allowing users to integrate computer vision and machine learning algorithms seamlessly.

Overall, OpenCV is a powerful and flexible library for computer vision and machine learning that provides a wide range of functions and tools for image and video processing. Its popularity and active development make it a valuable tool for researchers, engineers, and developers working in the field of computer vision.

3.6.2 PYTHON

Python is an interpreted, high-level, general-purpose programming language that utilizes imperative, functional, and object-oriented programming paradigms to provide a dynamic and modular approach to software development. Its design philosophy focuses on simplicity, readability, and ease of use, making it a popular choice for a wide range of applications, including web development, data science, artificial intelligence, and machine learning. With a rich ecosystem of libraries and frameworks, Python provides developers with the flexibility and tools necessary to efficiently build robust and reliable software solutions.

3.6.3 VS CODE

Visual Studio Code is a sophisticated integrated development environment that boasts an extensive suite of powerful functionalities and features, including advanced debugging tools, customizable user interface, code editing capabilities, syntax highlighting, and a vast array of extensions and plugins that augment its extensive range of capabilities. Moreover, its adaptable and flexible design enables it to accommodate various programming languages and frameworks, thereby enabling developers to undertake complex development projects with a high degree of precision and efficiency. Additionally, its ability to integrate seamlessly with various software development life cycle tools and package managers make it a highly sought-after tool among developers and software engineers

3.6.4 ANACONDA PROMPT

Anaconda Prompt is a command-line interface for managing and working with environments created by the Anaconda distribution of Python. Anaconda is a popular distribution of the Python programming language, which includes a package manager, a collection of pre-built packages, and other tools for scientific computing and data analysis.

Anaconda Prompt provides a command-line interface that allows users to manage environments, install packages, and execute Python code. It includes a set of pre-defined commands that can be used to manage environments and packages, such as creating new environments, activating and deactivating environments, installing and updating packages, and listing installed packages.

In addition to managing environments and packages, Anaconda Prompt can also be used to run Python scripts and execute Python code. It includes the Python interpreter, as well as other tools and libraries for scientific computing and data analysis, such as NumPy, SciPy, and pandas.

CHAPTER 4
SYSTEM DESIGN

CHAPTER 4
SYSTEM DESIGN

4.1 ER DIAGRAM

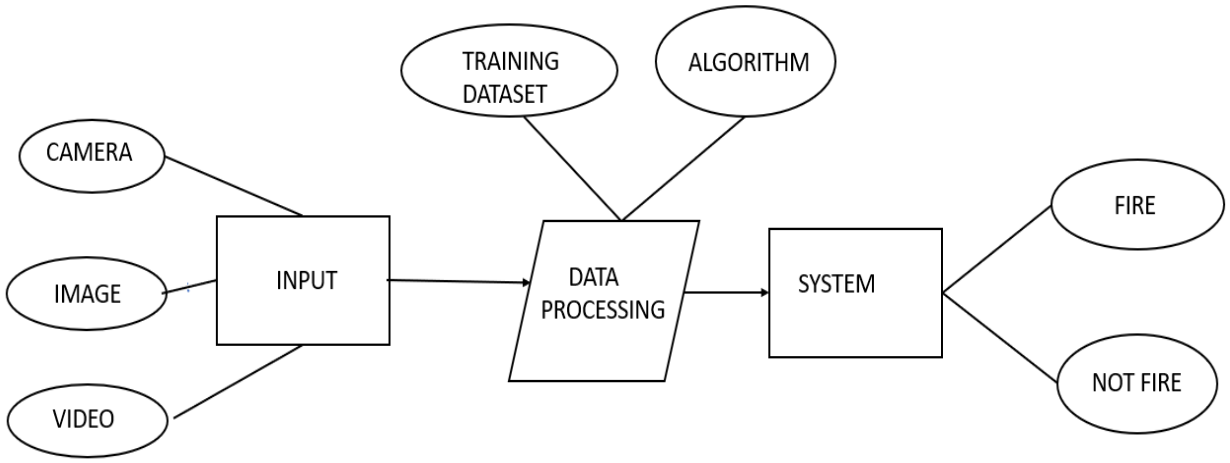


Fig.1 ER Diagram

EXPLANATION:

An entity relationship diagram (ERD), also known as an entity relationship model, is a graphical representation that depicts relationships among people, objects, places, concepts or events within an information technology (IT) system

4.2 DATA FLOW DIAGRAM

4.2.1 DATA FLOW DIAGRAM LEVEL 0

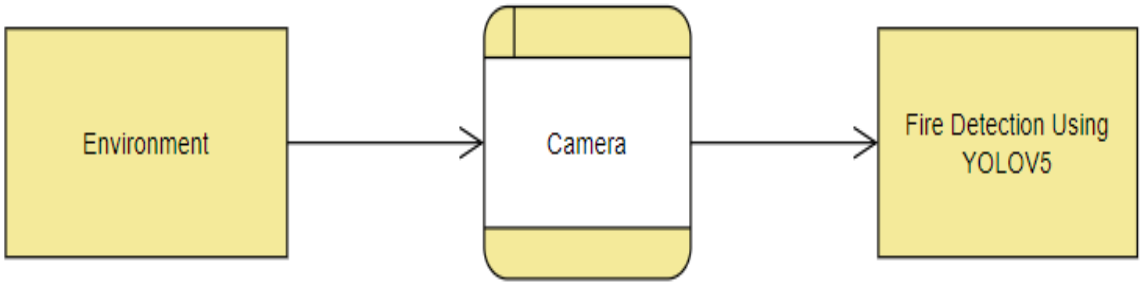


Fig.2 DFD level 0

EXPLANATION:
The Level 0 DFD shows how the system is divided into 'sub-systems' (processes), each of which deals with one or more of the data flows to or from an external agent, and which together provide all of the functionality of the system as a whole. It also identifies internal data stores that must be present in order for the system to do its job, and shows the flow of data between the various parts of the system.

4.2.2 DATA FLOW DIAGRAM LEVEL 1

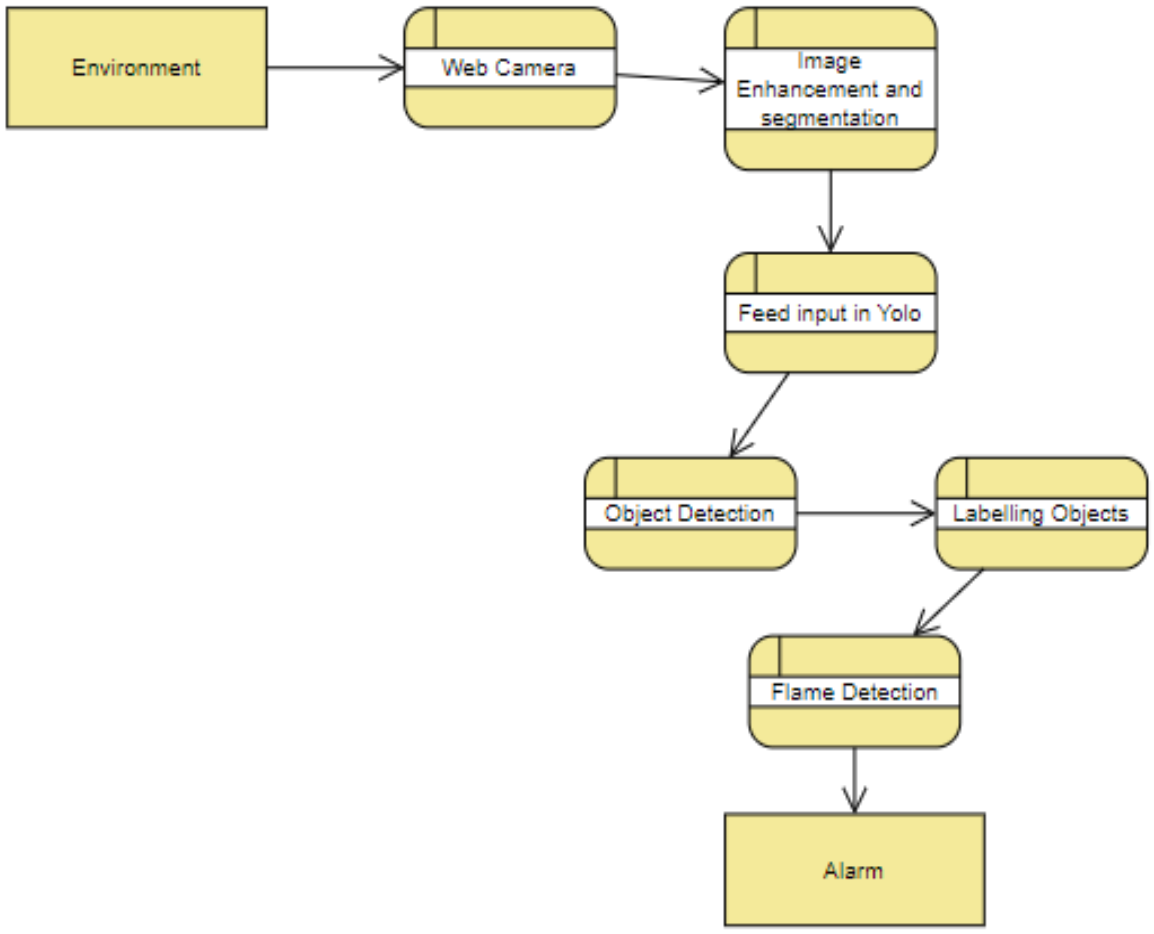


Fig.3 DFD level 1

EXPLANATION:

The next stage is to create the Level 1 Data Flow Diagram. This highlights the main functions carried out by the system. As a rule, to describe the system was using between two and seven functions - two being a simple system and seven being a complicated system. This enables us to keep the model manageable on screen or paper.

4.3 UML DIAGRAMS

4.3.1 USECASE DIAGRAM

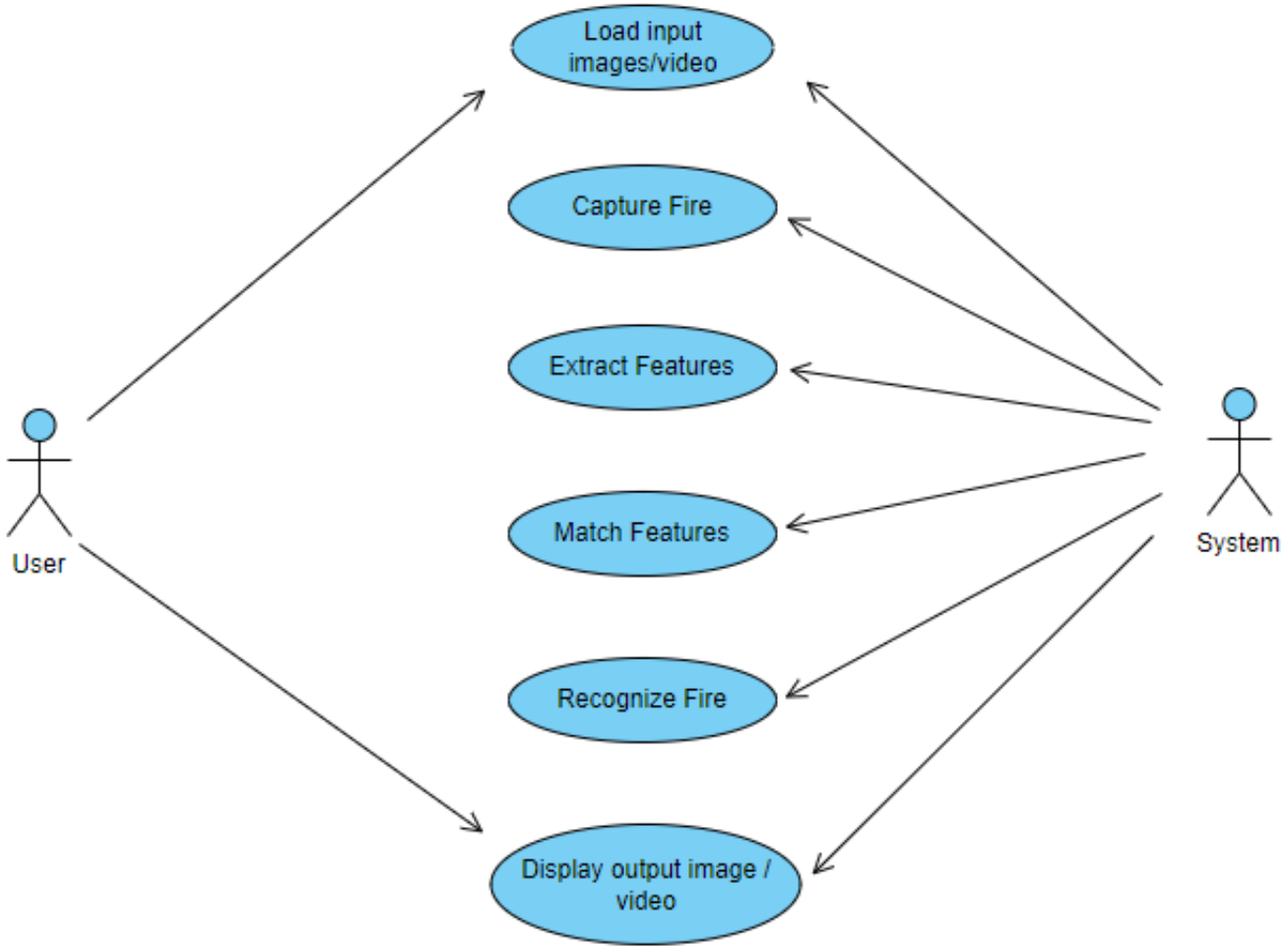


Fig.4 Usecase Diagram

EXPLANATION:

Use-case diagrams describe the high-level functions and scope of a system. These diagrams also identify the interactions between the system and its actors. The use cases and actors in use-case diagrams describe what the system does and how the actors use it, but not how the system operates internally.

4.3.2 SEQUENCE DIAGRAM

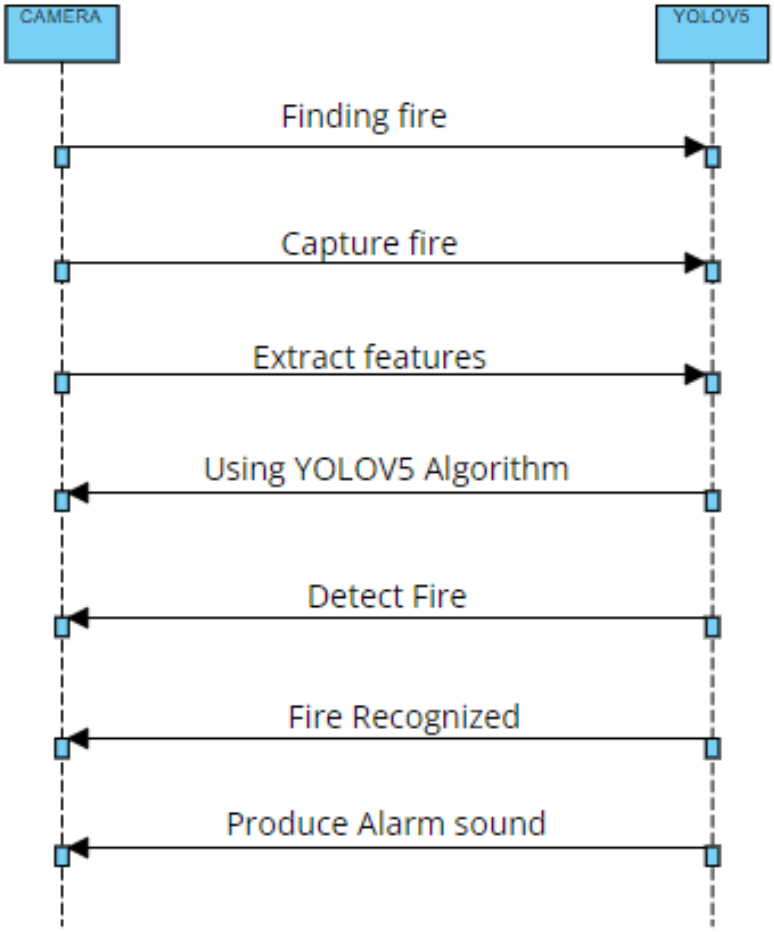


Fig.5 Sequence Diagram

EXPLANATION:

A sequence diagram is a type of interaction diagram because it describes how— and in what order—a group of objects works together. These diagrams are used by software developers and business professionals to understand requirements for a new system or to document an existing process.

4.3.3 ACTIVITY DIAGRAM

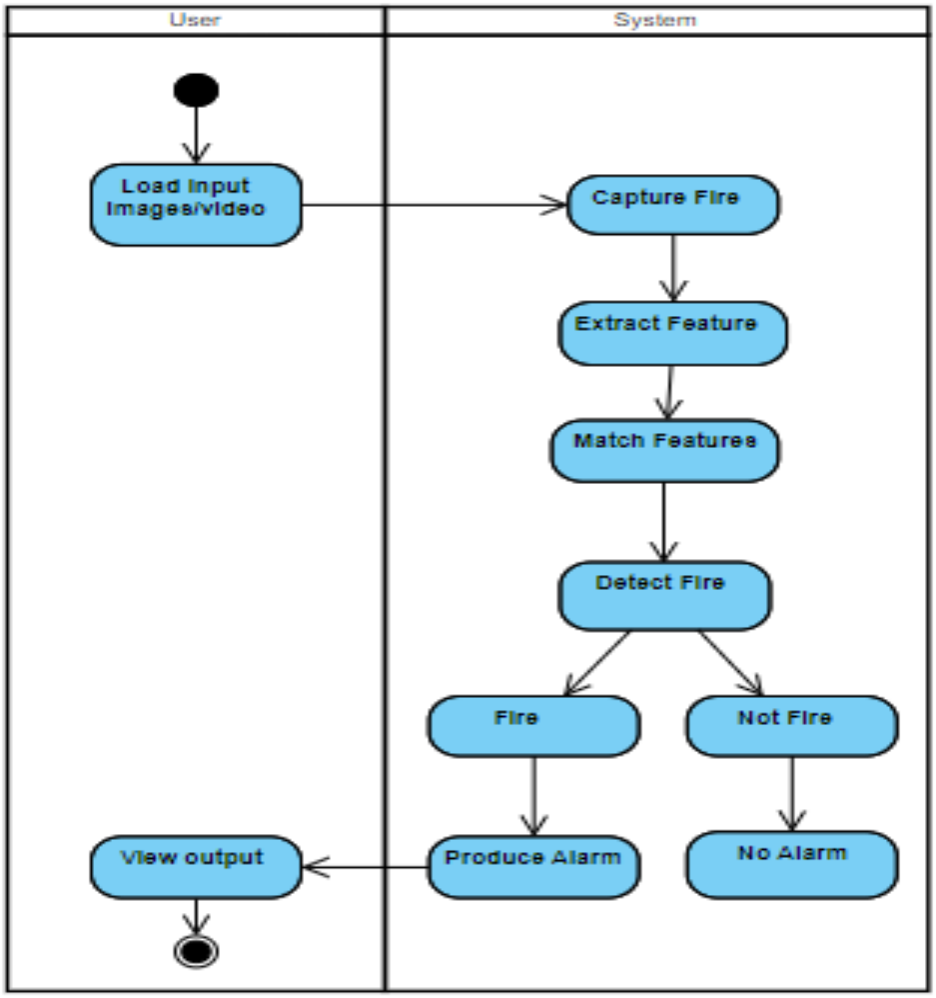


Fig.6 Activity Diagram

EXPLANATION:

An activity diagram visually presents a series of actions or flow of control in a system similar to a flowchart or a data flow diagram. Activity diagrams are often used in business process modeling. They can also describe the steps in a use case diagram.

4.4 ARCHITECTURE DIAGRAM

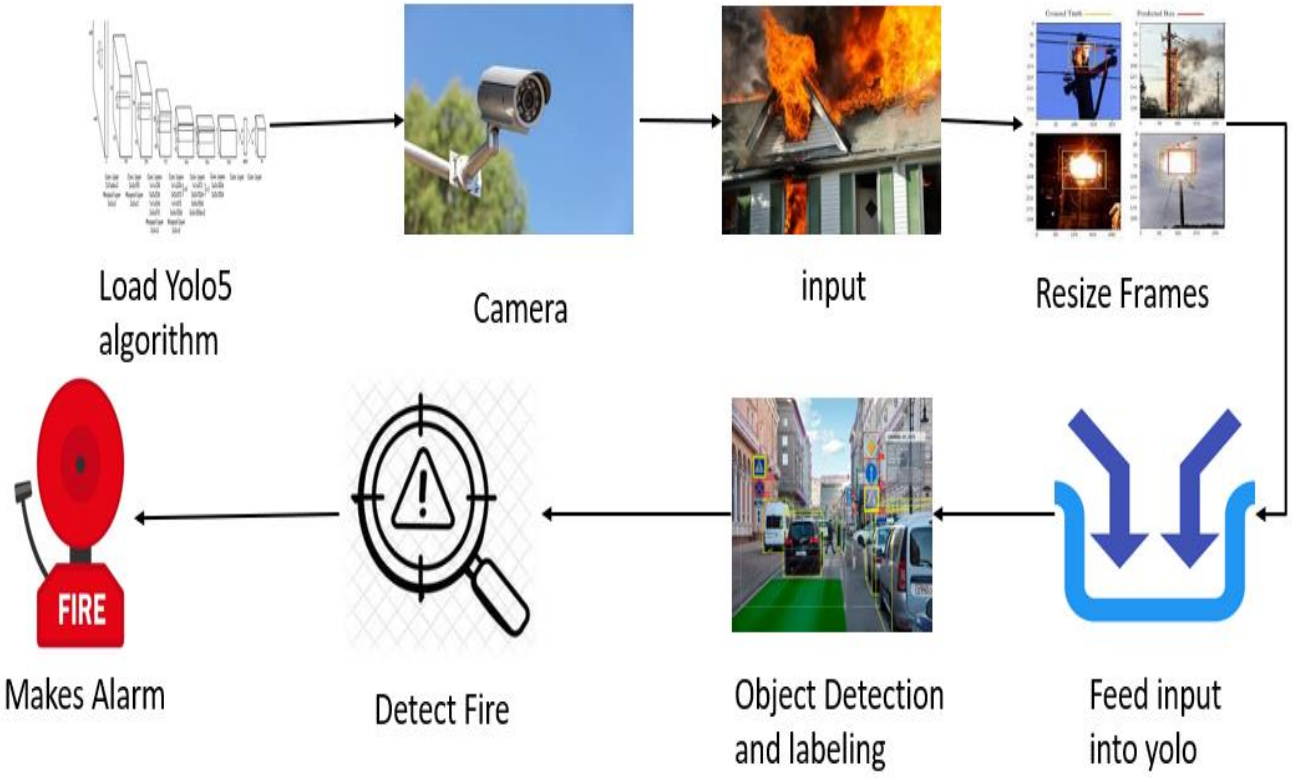


Fig.7 Architecture Diagram

CHAPTER 5
SYSTEM ARCHITECTURE

CHAPTER 5
SYSTEM ARCHITECTURE

5.1 **MODULE DESIGN SPECIFICATION**

1. LOAD YOLO.

1.1 Download Pre-trained Weights: The pre-trained weights for various YOLO versions are available on the official website. Download the weight files and store them in a folder
Official website: https://pjreddie.com/darknet/yolo/

1.2 Load the configuration and weight files: In OpenCV, you can load the configuration and weight files using the cv2.dnn.readNetFromDarknet() function. This function takes two arguments: the path to the configuration file and the path to the weight file.

```
net = cv2.dnn.readNetFromDarknet('yolov3.cfg', 'yolov3.weights')
```

1.3 The YOLO algorithm is trained to recognize a specific set of objects. The class names can be loaded by reading a text file that contains one class name per line.

```
with open('coco.names', 'r') as f:
    classes = [line.strip() for line in f.readlines()]
```

1.4 The input image needs to be preprocessed before feeding it to the YOLO network. Resizing the image to the same size as the input size of the network must be done.

```
image = cv2.imread('input_image.jpg')
resized_image = cv2.resize(image, (416, 416))
```

1.5 Once the input image is preprocessed, set it as the input to the YOLO network using the net.setInput() function.

```
net.setInput(resized_image)
```

1.6 The forward pass of the network will generate a list of bounding boxes and class probabilities for the objects in the image.

```
outputs = net.forward()
```

1.7 Once you have the outputs from the network, you need to filter out the weak detections and non-maximum suppression to get the final set of bounding boxes and class probabilities

2. START WEBCAM.

To start the webcam in OpenCV, you can use the cv2.VideoCapture() function. This creates a Video Capture object with a device index of 0, which represents the default webcam. It sets the width and height of the frame to 640 and 480, respectively, and starts a while loop to read frames from the webcam, display them, and exit the loop when the 'q' key is pressed.

3. IMAGE PROCESSING

Image processing is an essential component of YOLO, as it involves preparing and preprocessing the input images for object detection.
The important steps in image processing involves:

- Set Input and Output Layers: To set the input and output layers, use the "blobFromImage()" function to pre-process the input image for the network.
- Resizing the input image to a fixed size: The YOLO model expects a fixed input size. Therefore, the input images are resized to a fixed size before being fed to the model.
- Converting the image to a format suitable for the model: The YOLO model expects the input image to be in RGB color format. Therefore, if the input image is in a different format, it is converted to RGB.
- Normalization: The pixel values in the input image are normalized to a range between 0 and 1. This step helps improve the accuracy and stability of the object detection model

4.OBJECT DETECTION

- Object detection is a fundamental task in computer vision that involves identifying the location and class of objects within an image or video. YOLOv5 is a state-of-the-art object detection algorithm that has gained significant popularity due to its speed and accuracy.
- The YOLOv5 algorithm uses a deep neural network to predict bounding boxes, object classes, and confidence levels for each detected object in real-time. The algorithm works by dividing the input image into a grid of cells and predicting the probability of an object being present within each cell. Bounding boxes are then predicted for each cell that exceeds a certain threshold.
- One of the main advantages of YOLOv5 is its speed. It can process up to 155 frames per second on a single GPU, which makes it suitable for real-time applications such as autonomous vehicles, surveillance cameras, and robotics. Additionally, YOLOv5 achieves state-of-the-art performance on many benchmark datasets, making it a reliable choice for object detection tasks.
- To implement object detection using YOLOv5, one can use pre-trained models from the official repository or train a custom model on a specific dataset. The implementation involves initializing the model architecture, loading the pre-trained weights or training the model on the dataset, and feeding the input image to the model to obtain predictions. The output predictions can then be visualized by drawing bounding boxes around the detected objects and labeling them with their corresponding class.
- Overall, YOLOv5 has revolutionized object detection and inspired various applications across different industries. Its speed, accuracy, and ease of use make it a popular choice for developers and researchers alike.

5.POST PROCESSING

Post-processing is an essential step in computer vision (CV) that involves analysing and refining the output of a computer vision algorithm. Post-processing is particularly important in CV applications that involve object detection and recognition, as the output of these algorithms may contain errors or inaccuracies.

Some of the common techniques used in post-processing for CV include:

- Non-maximum suppression: Non-maximum suppression is a technique used to eliminate duplicate detections of the same object. This involves identifying overlapping bounding boxes and selecting the one with the highest confidence score.

- Bounding box refinement: Bounding box refinement is used to adjust the location and size of bounding boxes to better fit the detected object. This can improve the accuracy of object detection and recognition.

- Post-classification analysis: Post-classification analysis involves analyzing the output of a classification algorithm to identify patterns or trends. This can help to improve the accuracy of object detection and recognition.

6. ALERTING

- Alerting in OpenCV (Open Source Computer Vision Library) is a technique used to notify users of specific events or occurrences detected in a video stream or image. Alerting is a crucial component of many computer vision applications, such as security systems, where alerts need to be triggered if an intruder is detected.
- Overall, alerting in OpenCV is an essential feature for many computer vision applications. The ability to trigger alerts based on specific events detected in a video stream or image is critical in applications such as surveillance, security, and monitoring systems.

5.2 ALGORITHMS

YOLO ALGORITHM:

Yolo is a state-of-the-art, real-time object detector. Continuous improvements have made it achieve top performances on two official object detection datasets: Pascal VOC (visual object classes) and Microsoft COCO (common objects in context) . The network architecture of Yolov5 is shown in Figure 4. There are three reasons why we choose Yolov5 as our first learner. Firstly, Yolov5 incorporated cross stage partial network (CSPNet) into Darknet, creating CSPDarknet as its backbone. CSPNet solves the problems of repeated gradient information in large-scale backbones, and integrates the gradient changes into the feature map, thereby decreasing the parameters and FLOPS (floating-point operations per second) of model, which not only ensures the inference speed and accuracy, but also reduces the model size. In fire detection task, detection speed and accuracy is imperative, and compact model size also determines its inference efficiency on resource-poor edge devices. Secondly, the Yolov5 applied path aggregation network (PANet) as its neck to boost information flow. PANet adopts a new feature pyramid network (FPN) structure with enhanced bottom-up path, which improves the propagation of low-level features. At the same time, adaptive feature pooling, which links feature grid and all feature levels, is used to make useful information in each feature level propagate directly to following subnetwork. PANet improves the utilization of accurate localization signals in lower layers, which can obviously enhance the location accuracy of the object. Thirdly, the head of Yolov5, namely the Yolo layer, generates 3 different sizes (18 × 18, 36 × 36, 72 × 72) of feature maps to achieve multi-scale prediction, enabling the model to handle small, medium, and big objects.

There are two types of object detection models : two-stage object detectors and single-stage object detectors. Single-stage object detectors (like YOLO) architecture are composed of three components: **Backbone**, **Neck** and a **Head** to make dense predictions as shown in the figure bellow.

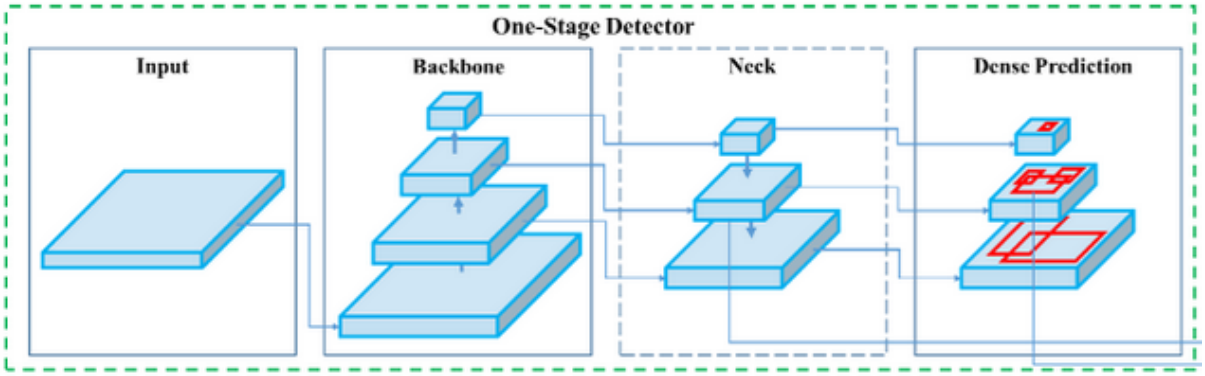


Fig.9 One state detector

MODEL BACKBONE

The backbone is a pre-trained network used to extract rich feature representation for images. This helps **reducing the spatial resolution** of the image and **increasing its feature** (channel) **resolution**.

MODEL NECK

The model neck is used to extract feature pyramids. This helps the model to generalize well to objects on different sizes and scales.

MODEL HEAD

The model head is used to perform the final stage operations. It applies anchor boxes on feature maps and render the final output: **classes** , **objectness scores** and **bounding boxes**.

YOLO ARCHITECTURE:

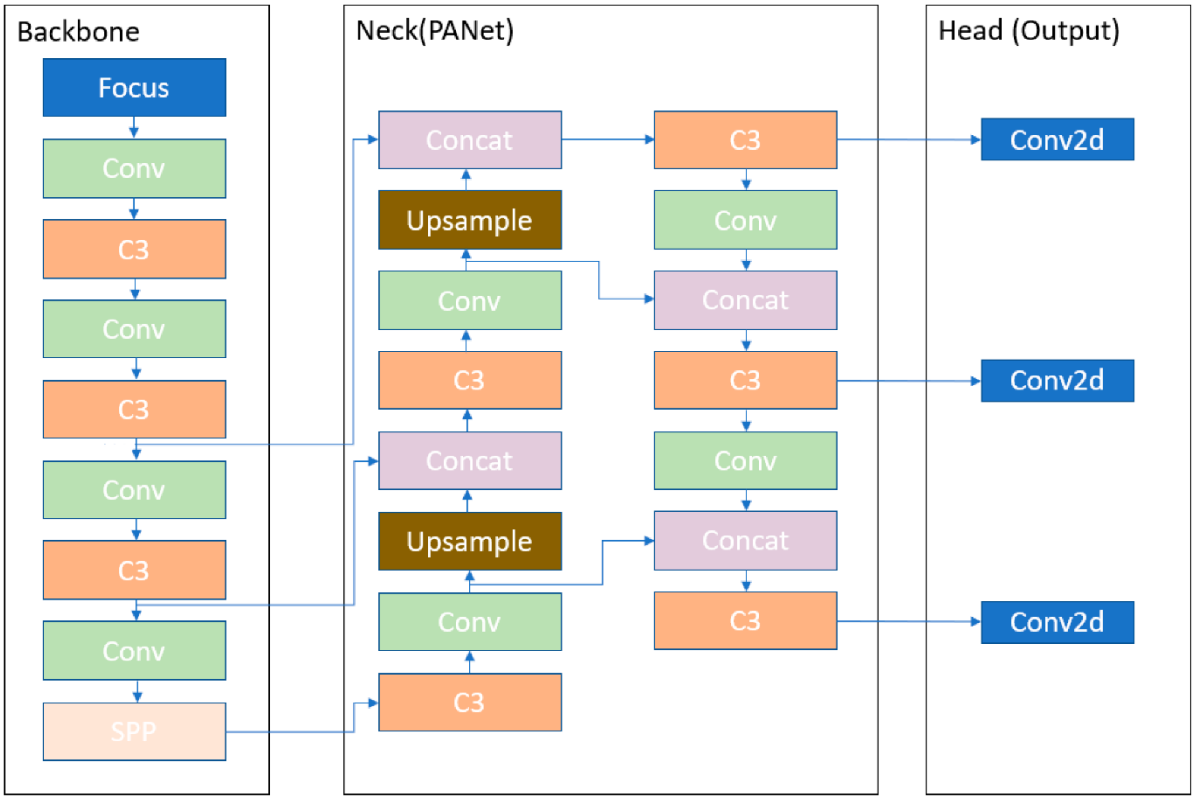


Fig.10 Architecture of yolo

All the YOLOv5 models are composed of the same 3 components: **CSP-Darknet53** as a backbone, **SPP** and **PANet** in the model neck and the **head**

Load pre-trained convolutional neural network (CNN)

while True:
    Load input image
    Resize input image to network size
    Feed input image through CNN
    Obtain output tensor of shape (S, S, B * (5 + C)), where:
        - S is the grid size (e.g. 7x7)
        - B is the number of bounding boxes per grid cell (e.g. 2)
        - C is the number of object classes
        - Each grid cell predicts B bounding boxes and their associated confidence scores and class probabilities

Interpret output tensor:

- For each grid cell, find the bounding box with the highest confidence score
- If the confidence score of the box is below a certain threshold, discard the box
- If the confidence score is above the threshold, consider the box a detection
- Apply non-max suppression to remove redundant detections

Draw bounding boxes around detected objects in the input image

Display output image with bounding boxes

CHAPTER 6
SYSTEM IMPLEMENTATION

CHAPTER 6
SYSTEM IMPLEMENTATION

6.1 CODING

```
import cv2
import numpy as np
import argparse
import time


parser = argparse.ArgumentParser()
parser.add_argument('--webcam', help="True/False", default=False)
parser.add_argument('--play_video', help="Tue/False", default=False)
parser.add_argument('--image', help="Tue/False", default=False)
parser.add_argument('--video_path', help="Path of video file",
default="videos/fire1.mp4")
parser.add_argument('--image_path', help="Path of image to detect objects",
default="Images/bicycle.jpg")
parser.add_argument('--verbose', help="To print statements", default=True)
args = parser.parse_args()


#Load yolo
def load_yolo():
    net = cv2.dnn.readNet("yolov3-tiny-obj_final.weights", "yolov3-tiny.cfg")
    classes = []
    with open("obj.names", "r") as f:
        classes = [line.strip() for line in f.readlines()]

    layers_names = net.getLayerNames()
    output_layers = [layers_names[i-1] for i in net.getUnconnectedOutLayers()]
    colors = np.random.uniform(0, 255,255)
    return net, classes, colors, output_layers

def load_image(img_path):
    # image loading
    img = cv2.imread(img_path)
```

```
    img = cv2.resize(img, None, fx=0.4, fy=0.4)
    height, width, channels = img.shape
    return img, height, width, channels

def start_webcam():
    cap = cv2.VideoCapture(0)

    return cap


def display_blob(blob):

    for b in blob:
        for n, imgb in enumerate(b):
            cv2.imshow(str(n), imgb)

def detect_objects(img, net, outputLayers):
    blob = cv2.dnn.blobFromImage(img, scalefactor=0.00392, size=(320, 320),
mean=(0, 0, 0), swapRB=True, crop=False)
    net.setInput(blob)
    outputs = net.forward(outputLayers)
    return blob, outputs

def get_box_dimensions(outputs, height, width):
    boxes = []
    confs = []
    class_ids = []
    for output in outputs:
        for detect in output:
            scores = detect[5:]
            class_id = np.argmax(scores)
            conf = scores[class_id]
            if conf > 0.3:
                center_x = int(detect[0] * width)
                center_y = int(detect[1] * height)
                w = int(detect[2] * width)
                h = int(detect[3] * height)
                x = int(center_x - w/2)
```

```
                        y = int(center_y - h / 2)
                        boxes.append([x, y, w, h])
                        confs.append(float(conf))
                        class_ids.append(class_id)
        return boxes, confs, class_ids

def draw_labels(boxes, confs, colors, class_ids, classes, img):
        indexes = cv2.dnn.NMSBoxes(boxes, confs, 0.5, 0.4)
        font = cv2.FONT_HERSHEY_PLAIN
        for i in range(len(boxes)):
                if i in indexes:
                        x, y, w, h = boxes[i]
                        label = str(classes[class_ids[i]])
                        color = colors[i]
                        cv2.rectangle(img, (x,y), (x+w, y+h), color, 2)
                        cv2.putText(img, label, (x, y - 5), font, 1, color, 1)
                        if(label=="Fire"):
                                dimg= start_webcam().read()[1]
                                cv2.imwrite('test.jpg', dimg)
                                import win32api
                                import random

                                for i in range(5):
                                        win32api.Beep(random.randint(37,10000),
random.randint(750,3000))
                                # mail.SendMail("test.jpg")
                                # webcam_detect()


        img=cv2.resize(img, (800,600))
        cv2.imshow("Image", img)

def image_detect(img_path):
        model, classes, colors, output_layers = load_yolo()
        image, height, width, channels = load_image(img_path)
        blob, outputs = detect_objects(image, model, output_layers)
        boxes, confs, class_ids = get_box_dimensions(outputs, height, width)
        draw_labels(boxes, confs, colors, class_ids, classes, image)
```

```
    while True:
        key = cv2.waitKey(1)
        if key == 27:
            break

def webcam_detect():
    model, classes, colors, output_layers = load_yolo()
    cap = start_webcam()
    while True:
        success, frame = cap.read()
        if success:
            height, width, channels = frame.shape
            blob, outputs = detect_objects(frame, model, output_layers)
            boxes, confs, class_ids = get_box_dimensions(outputs, height,
width)
            draw_labels(boxes, confs, colors, class_ids, classes, frame)
        key = cv2.waitKey(1)

        if key == 27:
            break
    cap.release()


def start_video(video_path):
    model, classes, colors, output_layers = load_yolo()
    cap = cv2.VideoCapture(video_path)

    while True:
        _, frame = cap.read()
        height, width, channels = frame.shape
        blob, outputs = detect_objects(frame, model, output_layers)
        boxes, confs, class_ids = get_box_dimensions(outputs, height, width)
        draw_labels(boxes, confs, colors, class_ids, classes, frame)

        key = cv2.waitKey(1)
        if cv2.waitKey(1) & 0xFF ==ord('q'):
            break
    cap.release()
```

```
if __name__ == '__main__':
        webcam = args.webcam
        video_play = args.play_video
        image = args.image
        if webcam:
                if args.verbose:
                        print('---- Starting Web Cam object detection ----')
                webcam_detect()
        if video_play:
                video_path = args.video_path
                if args.verbose:
                        print('Opening '+video_path+" .... ")
                start_video(video_path)
        if image:
                image_path = args.image_path
                if args.verbose:
                        print("Opening "+image_path+" .... ")
                image_detect(image_path)


        cv2.destroyAllWindows()
```

CHAPTER 7
PERFORMANCE ANALYSIS

CHAPTER 7
PERFORMANCE ANALYSIS

SYSTEM TESTING AND MAINTENANCE:

Testing is vital to the success of the system. System testing makes a logical assumption that if all parts of the system are correct, the goal will be successfully achieved. In the testing process we test the actual system in an organization and gather errors from the new system operates in full efficiency as stated. System testing is the stage of implementation, which is aimed to ensuring that the system works accurately and efficiently. In the testing process we test the actual system in an organization and gather errors from the new system and take initiatives to correct the same. All the front-end and back-end connectivity are tested to be sure that the new system operates in full efficiency as stated. System testing is the stage of implementation, which is aimed at ensuring that the system works accurately and efficiently.

The main objective of testing is to uncover errors from the system. For the uncovering process we have to give proper input data to the system. So we should have more conscious to give input data. It is important to give correct inputs to efficient testing.Testing is done for each module. After testing all the modules, the modules are integrated and testing of the final system is done with the test data, specially designed to show that the system will operate successfully in all its aspects conditions. Thus, the system testing is a confirmation that all is correct and an opportunity to show the user that the system works. Inadequate testing or non-testing leads to errors that may appear few months later.This will create two problems, Time delay between the cause and appearance of the problem. The effect of the system errors on files and records within the system. The purpose of the system testing is to consider all the likely variations to which it will be suggested and push the system to its limits.

The testing process focuses on logical intervals of the software ensuring that all the statements have been tested and on the function intervals (i.e.,) conducting tests to uncover errors and ensure that defined input will produce actual results that agree with the required results. Testing has to be done using the two common steps Unit testing and Integration testing. In the project system testing is made as follows: The procedure level testing is made first. By giving improper inputs, the errors occurred are noted and eliminated. This is the final step in system life cycle. Here we implement the tested

error-free system into real-life environment and make necessary changes, which runs in an online fashion. Here system maintenance is done every months or year based on company policies, and is checked for errors like runtime errors, long run errors and other maintenances like table verification and reports. Integration Testing is a level of software testing where individual units are combined and tested as a group.

7.1 UNIT TESTING

Unit testing verification efforts on the smallest unit of software design, module. This is known as "Module Testing". The modules are tested separately. This testing is carried out during programming stage itself. In these testing steps, each module is found to be working satisfactorily as regard to the expected output from the module.

BLACK BOX TESTING

Black box testing, also known as Behavioral Testing, is a software testing method in which the internal structure/ design/ implementation of the item being tested is not known to the tester. These tests can be functional or non-functional, though usually functional.

WHITE-BOX TESTING

White-box testing (also known as clear box testing, glass box testing, transparent box testing, and structural testing) is a method of testing software that tests internal structures or workings of an application, as opposed to its functionality (i.e. black-box testing).

GREY BOX TESTING

Grey box testing is a technique to test the application with having a limited knowledge of the internal workings of an application. To test the Web Services application usually the Grey box testing is used. Grey box testing is performed by end-users and also by testers and developers.

7.2 INTEGRATION TESTING

Integration testing is a systematic technique for constructing tests to uncover error associated within the interface. In the project, all the modules are combined and then

the entire programmer is tested as a whole. In the integration-testing step, all the error uncovered is corrected for the next testing steps. Software integration testing is the incremental integration testing of two or more integrated software components on a single platform to produce failures caused by interface defects. The task of the integration test is to check that components or software applications, e.g. components in a software system or – one step up – software applications at the company level – interact without error.

7.3 TEST CASES AND REPORTS

S.NO	TEST CASE	INPUT	STEPS	OUTPUT
1	Fire detection accuracy	Different types of fires and non-fire images.	a. Create a dataset of images and videos containing different types of fires and non-fire objects b. Use the YOLOv5 algorithm to detect the presence of fire in the images and videos in the dataset c. Record the number of true positive, false positive, true negative, and false negative detections d. Calculate the precision, recall, and F1 score to evaluate the accuracy of fire detection using YOLOv5.	Correct prediction of images.
2	YOLOv5 performance on different types of fires	Images resembling different types of fires.	a. Create a dataset of images and videos containing different types of fires such as campfires, wildfires, and house fires b. Use the YOLOv5 algorithm to detect the presence of fire in the images and videos in the dataset c. Compare the	Correct prediction of images even the location, size, and type of fire changes. The prediction is accurate

			performance of YOLOv5 in detecting different types of fires by analyzing the accuracy and detection time.	
3	Impact of environmental factors on YOLOv5's fire detection accuracy	Live video in different lighting conditions through web cam or images with different lighting conditions.	a. Test YOLOv5's fire detection performance under different lighting conditions such as daylight, low light, and darkness b. Observe how the algorithm performs in detecting fires under different weather conditions, such as haze or smoke. c. Measure how well YOLOv5 works in detecting fires at varying distances.	The system predicts the fire accurately.
4	YOLOv5's ability to detect false positives	Images and videos containing objects that resemble fire.	a. Create a dataset of images and videos containing objects that resemble fire, such as bright lights, reflections, and sunsets. b. Run YOLOv5 algorithm to detect the presence of a fire in the images and videos in the dataset. c. Analyze the number of false positives generated by YOLOv5 by	Detects only fire and not images that resemble fire.

			examining the images and videos in the dataset.	
5	YOLOv5's ability to detect fires in crowded scenes	Fire images/video occurring in a crowded location.	a. Create a dataset of images containing crowds of people in which a fire occurs b. Run YOLOv5 algorithm to detect the presence of a fire in the images in the dataset. c. Measure how well the algorithm works in detecting fires when there is a crowd of people and other obstacles in the scene.	Detect fire even in a crowded place and produce alarm sound.

1. Test cases and reports

CHAPTER 8
CONCLUSION

CHAPTER 8
CONCLUSION

8.1 RESULTS AND DISCUSSIONS

A crucial area of research entails creating algorithms and systems that can recognise fires as soon as they start. This is known as real-time fire detection. Real-time fire detection aims to minimise possible fire damage, such as property loss, human fatalities, and environmental harm. The system was designed in such a way that successful fire detection could be accomplished because fire detection is crucial and must be done at the appropriate moment. The system sounds a warning as soon as a fire is discovered to alert people about the fire. A person nearby can be reached and the fire can be put out at the earliest stage. Additionally, it can be used to keep an eye on commercial buildings and spot possible fire hazards early on, enabling early intervention and prevention. In conclusion, real-time fire detection is a vital tool with the potential to prevent deaths and lessen the harm that fires do. Fire detection is successful with the growth of this system, making it a crucial component of fire prevention and control.

8.2 CONCLUSION AND FUTURE ENHANCEMENTS

8.2.1 CONCLUSION

The fundamental structure of YOLOV5 is real-time object detection algorithm. When properly used, YOLOV5 has many advantages in practice. Being a unified object detection model that is simple to construct and train in correspondence with its simple loss-function, YOLOv5 can train the entire model in parallel. YOLOV5 is also better at generalizing object representation compared with other object detection models and can be recommended for real-time object detection as the state-of-art algorithm in fire detection. With these marks, it is acknowledgeable that the field of fire detection has an expanding, great future ahead. The experiments showed that changing the algorithm (model size) and the dataset could quickly detect a real-time fire with a high degree of accuracy.

8.2.2 FUTURE ENHANCEMENT

- Future work will include improving the accuracy of our method and handling blurring issues in nighttime environments.
- Our future projection is to build a lightweight model with robust detection performance that would allow us to set up embedded devices with low computational capabilities.
- Future work will be based on using geostationary satellite imagery for rapid monitoring of active fires to provides high temporal resolution for fast monitoring in a larger scale.

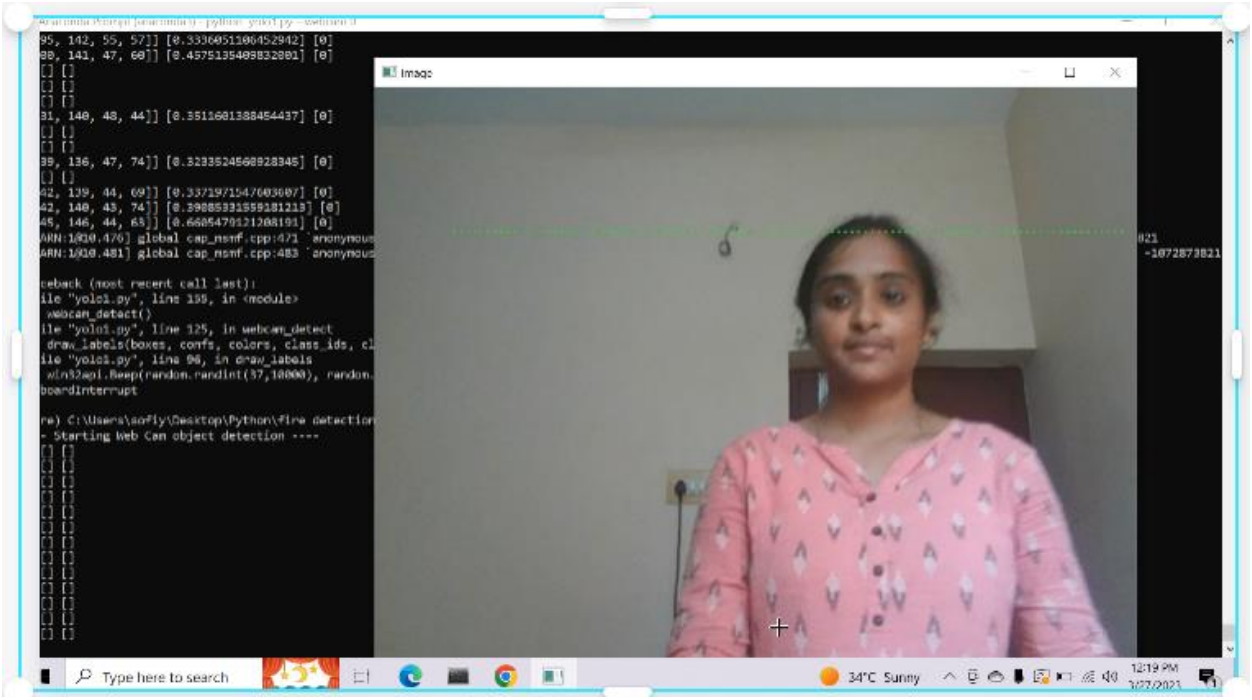
CHAPTER-9
APPENDICES

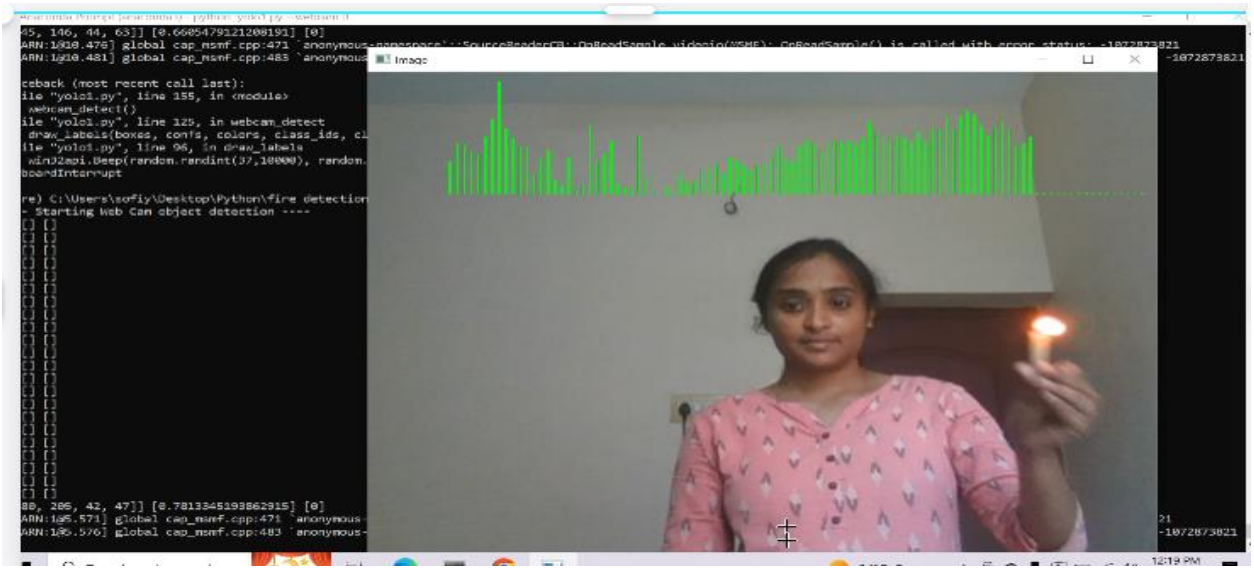
CHAPTER-9
APPENDICES

9.1 SAMPLE SCREENS

INPUT:

OUTPUT:




45, 146, 44, 63]] [0.6605479121208191] [0]
ARN:1@10.476] global cap_msmf.cpp:471 `anonymous-
ARN:1@10.481] global cap_msmf.cpp:483 `anonymous
ceback (most recent call last):
ile "yolo1.py", line 155, in <module>
webcam_detect()
ile "yolo1.py", line 125, in webcam_detect
draw_labels(boxes, confs, colors, class_ids, cl
ile "yolo1.py", line 96, in draw_labels
winsound.Beep(random.randint(37,10000), random.
boardInterrupt
re) C:\Users\sofiy\Desktop\Python\fire detection
- Starting Web Cam object detection ----
88, 286, 42, 47]] [0.7813345193862915] [0]
ARN:1@5.571] global cap_msmf.cpp:471 `anonymous-
ARN:1@5.576] global cap_msmf.cpp:483 `anonymous-
Image
-1072873821
-1072873821

9.2 PLAGERISM REPORT:

ORIGINALITY REPORT

29 %
SIMILARITY INDEX

PRIMARY SOURCES

1 www.researchgate.net
Internet

2 www.mdpi.com
Internet

3 fireecology.springeropen.com
Internet

4 www.ncbi.nlm.nih.gov
Internet

5 Utkarsh Meena, Geetika Munjal, Sanya Sachdeva, Pranjul Garg, Daksh Dagar, Anushka Gangal. "RCNN

424 words — 8%

128 words — 2%

294 words — 5%

104 words — 2%

93 words — 2%

Architecture for Forest Fire Detection", 2023 13th International Conference on Cloud Computing, Data Science & Engineering (Confluence), 2023

Crossref

6 Xuehui Wu, Xiaobo Lu, Henry Leung. "An adaptive threshold deep learning method for fire and smoke detection", 2017 IEEE International Conference on Systems, Man, and Cybernetics (SMC), 2017

85 words — 2%

Crossref

7 K. Revathi, T. Tamilselvi, R. Arunkumar, T. Divya."Spot Fire: An Intelligent Forest Fire Detection System Design With Machine Learning", 2022 Internationational

55 words — 1%

Crossref

8 www.hindawi.com

Internet

53 words — 1%

49 words — 1%

9 doaj.org

Internet

35 words — 1%

30 words — 1%

10 jiki.cs.ui.ac.id

Internet

24 words — < 1%

11 www.ruivaledesousa.com

Internet

12 Shuai Zhao, Boyun Liu, Zheng Chi, Taiwei Li, Shengnan Li. "Characteristics Based Fire Detection System Under the Effect of Electric Fields With Improved Yolo-v4 and ViBe", IEEE Access, 2022

Crossref

13 Byoungjun Kim, Joonwhoan Lee. "A Bayesian Network-Based Information Fusion Combined with DNNs for Robust Video Fire Detection", Applied Sciences, 2021

23 words — < 1%

Crossref

14 origin.geeksforgeeks.org
Internet
16 words — < 1%

13 words — < 1%

15 www.nature.com
Internet
11 words — < 1%

16 Veerappampalayam Easwaramoorthy Sathishkumar, Jaehyuk Cho, Malliga Subramanian, Obuli Sai Naren. "Forest fire and smoke

detection using deep learning-based learning withoutforgetting", Fire Ecology, 2023
Crossref

17 www.jssec.org
Internet
11 words — < 1%

9 words — < 1%

18 Saurav Kumar, Sarthak Malik, P. Sumathi. "Deep Learning-based Border Surveillance System using Thermal Imaging", 2022 IEEE 19th India Council International Conference (INDICON), 2022
Crossref

19 Zhenyang Xue, Haifeng Lin, Fang Wang. "A Small Target Forest Fire Detection Model Based on YOLOv5 Improvement", Forests, 2022
Crossref
9 words — < 1%

Zubair Khaliq, Dawood Ashraf Khan, Sheikh Umar Farooq. "Using deep learning for selenium web UI

20 functional tests: A case-study with e-commerce applications", Engineering Applications of Artificial Intelligence, 2023
Crossref
9 words — < 1%

21 22 23

doc.lagout.org

Internet

9 words — < 1%

24

9 words — < 1%

problemsolvingwithpython.com

Internet

8 words — < 1%

Akhila Kambhatla, Khaled R Ahmed. "Chapter 13 Firearm Detection Using Deep Learning", Springer Science and Business Media LLC, 2023

Crossref

8 words — < 1%

Murat Muhammet Savci, Yasin Yildirim, Gorkem Saygili, Behcet Ugur Toreyin. "Fire Detection in H.264 Compressed Video", ICASSP 2019 - 2019 IEEE

International Conference on Acoustics, Speech and SignalProcessing (ICASSP), 2019

Crossref

---

25 Samia Choueiri, Daoud Daoud, Samir Harb, Roger Achkar. "Fire and Smoke Detection Using Artificial Neural Networks", 2020 14th International Conference on Open Source Systems and Technologies (ICOSST), 2020

8 words — < 1%

Crossref

---

26 hal.mines-ales.fr

8 words — < 1%

Internet

---

8 words — < 1%

---

27 www.researchsquare.com

7 words — < 1%

Internet

---

28 Akshaya Kumar Mandal. "Usage of Particle Swarm Optimization in Digital Images Selection for Monkeypox Virus Prediction and Diagnosis", Research Square Platform LLC, 2022

Crossref Posted Content

---

29 Yuming Li, Wei Zhang, Yanyan Liu, Rudong Jing, Changsong Liu. "An efficient fire and smoke detection algorithm based on an end-to-end structured network", Engineering Applications of Artificial Intelligence,2022

7 words — < 1%

Crossref

CHAPTER 10
REFERENCES

CHAPTER 10
REFERENCES

[1] Veerappampalayam Easwaramoorthy Sathishkumar, Jaehyuk Cho, Malliga Subramanian, and Obuli Sai Naren, “Forest fre and smoke detection using deep learning-based learning without forgetting”, 2023.

[2] Seyd Teymoor Seydi, Vahideh Saeidi, Bahareh Kalantar, Naonori Ueda and Alfian Abdul Halin, "Fire-Net: A Deep Learning Framework for Active Forest Fire Detection", Feb 2022

[3] Shuai Zhao, Boyun Liu, Zheng Chi, Taiwei Li and Shengnan Li, "Characteristics Based Fire Detection System,Under the Effect of Electric Fields With Improved Yolo-v4 and ViBe", August 2022.

[4] Byoungjun Kim and Joonwhoan Lee, “A Video-Based Fire Detection Using Deep Learning Models”, July 2019.

[5] Herminarto Nugroho, "Fully Convolutional Variational Autoencoder For Feature Extraction Of Fire Detection System", March 2020.

[6] Xiwen Chen, Bryce Hopkins, Hao Wang, Leo O' Neill, Fatemeh Afghah, et al, "Wildland Fire Detection and Monitoring Using a Drone-Collected RGB/IR Image Dataset", Nov 2022.

[7] Xuehui Wu1, Xiaobo Lu1, and Henry Leung, “An Adaptive Threshold Deep Learning Method for Fire and Smoke Detection,” Oct 2017

[8] Taha Zaman, Muhammad Hasan, Saneeha Ahmed, Shumaila Ashfaq, "Fire Detection Using Computer Vision" , 2019

[9] Murat Muhammet Savcı, Yasin Yıldırım, Gorkem Saygılı, and Behcet Ugur Toreyin, “Fire Detection in H.264 Compressed Video” 2019.

[10] Xuehui Wu, Xiaobo Lu, Henry Leung, "An Adaptive Threshold Deep Learning Method for Fire and Smoke Detection", Oct 2017.

[11] Zhenyang Xue, Haifeng Lin, and Fang Wang, “A Small target forest fire detection model based on yolov v5 improvement” Aug 2022.