

FIRE FIGHTER ROBOT WITH DEEP LEARNING AND MACHINE VISION

Submitted in partial fulfillment of the
requirements for the award of
Bachelor of Engineering degree in Computer Science and Engineering

By

Katepalli Deva Naga Sai Kumar (Register No. 40110570)

Kasireddy Krishna Madhava Reddy (Register No. 40110566)



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

SCHOOL OF COMPUTING

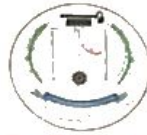
SATHYABAMA

**INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)**

CATEGORY -1 UNIVERSITY BY UGC

**Accredited with Grade “A++” by NAAC | Approved by AICTE
JEPPIAAR NAGAR, RAJIV GANDHI SALAI, CHENNAI - 600 119**

APRIL - 2024



SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY

(DEEMED TO BE UNIVERSITY)

Category - I University by UGC

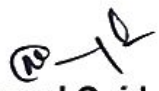
Accredited "A++" by NAAC | Approved by AICTE

www.sathyabama.ac.in

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

BONAFIDE CERTIFICATE

This is to certify that this Project Report is the bonafide work of **Kasireddy Krishna Madhava Reddy (40110566)** and **Katepalli Deva Naga Sai Kumar (40110570)** who carried out the Project entitled "**FIRE FIGHTER ROBOT WITH DEEP LEARNING AND MACHINE VISION**" under my supervision from November 2023 to April 2024.

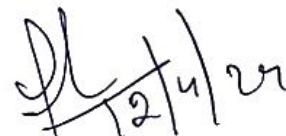

Internal Guide

Dr. NANCY NOELLA R.S, M.E., Ph.D.


HEAD OF THE DEPARTMENT
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
Dr. L. DEEPAK SATHYANARAYANA, Ph.D.
Jeppiaar Nagar, Rajiv Gandhi Salai,
Chennai - 600 119.

Submitted for Viva voce Examination held on 12-04-2024


Internal Examiner 12/4/24

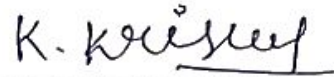

External Examiner

DECLARATION

I, **Kasireddy Krishna Madhava Reddy (40110566)** hereby declare that the Project Report entitled **"FIRE FIGHTING ROBOT WITH DEEP LEARNING AND MACHINE VISION"** done by me under the guidance of **Dr. NANCY NOELLA R.S, M.E., Ph.D.** is submitted in partial fulfillment of the requirements for the award of Bachelor of Engineering degree in **Computer Science and Engineering**.

DATE: 10-04-2024

PLACE: Chennai



SIGNATURE OF THE CANDIDATE

ACKNOWLEDGEMENT

I am pleased to acknowledge my sincere thanks to **Board of Management of SATHYABAMA** for their kind encouragement in doing this project and for completing it successfully. I am grateful to them.

I convey my thanks to **Dr. T. Sasikala M.E., Ph.D.**, Dean, School of Computing, **Dr. L. Lakshmanan M.E., Ph.D.**, Head of the Department of Computer Science and Engineering for providing me necessary support and details at the right time during the progressive reviews.

I would like to express my sincere and deep sense of gratitude to my Project Guide **Dr. NANCY NOELLA R.S, M.E., Ph.D.** for her valuable guidance, suggestions and constant encouragement paved way for the successful completion of my project work.

I wish to express my thanks to all Teaching and Non-teaching staff members of the **Department of Computer Science and Engineering** who were helpful in many ways for the completion of the project.

ABSTRACT

The project work is based on a groundbreaking advancement in the realm of fire detection within the domain of intelligent robotic systems. Utilizing cutting-edge deep learning algorithms, notably YOLOv8, we have successfully integrated and refined a methodology that delivers exceptional accuracy in identifying instances of fire within camera imagery. Our approach boasts an impressive 94.5% accuracy rate, underscoring a significant leap forward in the domain of real-time object recognition.

Recognizing the significance of this achievement, our research also identifies avenues for further enhancement, particularly in the nuanced differentiation of diverse fire types. By leveraging the prowess of machine vision technologies within a sophisticated robotic framework, our methodology lays a robust groundwork for adaptability in responding to a wide array of fire scenarios, ranging from conventional flames to more complex fire phenomena.

Beyond its immediate implications, this milestone signifies a pivotal step towards more efficient and effective fire response mechanisms. Moreover, it underscores the ongoing convergence of deep learning and robotics, marking a transformative juncture in the evolution of intelligent systems. The practical applications of our findings extend far beyond academia, holding immense promise for deployment in real-world emergency response and public safety contexts.

The results presented herein, represent a significant achievement with far-reaching implications. By enhancing fire detection capabilities through the integration of advanced technologies, we pave the way for safer and more resilient communities, epitomizing the potential of intelligent robotics in addressing critical challenges in disaster management and mitigation efforts.

TABLE OF CONTENTS

Chapter No.	TITLE	Page No.
	ABSTRACT	v
	LIST OF FIGURES	viii
	LIST OF ABBREVIATIONS	ix
1	INTRODUCTION	1
2	LITERATURE SURVEY	2
	2.1 INFERENCES FROM LITERATURE SURVEY	5
	2.2 OPEN PROBLEMS IN EXISTING SYSTEM	6
3	REQUIREMENT ANALYSIS	7
	3.1 HARDWARE REQUIREMENTS	7
	3.2 SOFTWARE REQUIREMENTS	7
	3.2.1 OPERATING SYSTEM	7
	3.2.2 PROGRAMMING LANGUAGES	8
	3.2.3 LIBRARIES AND FRAMEWORKS	8
	3.2.4 ALGORITHMS AND TECHNIQUES	9
	3.2.5 DATA STORAGE AND MANAGEMENT	9
	3.3 DATA COLLECTION	9
4	DESCRIPTION OF PROPOSED SYSTEM	11
	4.1 AIM	11
	4.2 SCOPE	11
	4.3 ARCHITECTURE / OVERALL DESIGN OF PROPOSED SYSTEM	12
	4.4 UML DIAGRAMS FOR FIREFIGHTING ROBOT	12
	4.4.1 USECASE DIAGRAM	13
	4.4.2 CLASS DIAGRAM	13
	4.4.3 DATA FLOW LEVEL 0 DIAGRAM	14
	4.4.4 DATA FLOW LEVEL 1 DIAGRAM	14

	4.4.5	SEQUENCE DIAGRAM	15
	4.4.6	STATE CHART DIAGRAM	16
	4.5	MATHEMATICAL MODEL	16
5		IMPLEMENTATION OF PROPOSED SYSTEM	18
	5.1	WORKING PROCESS	18
	5.2	FIRE DETECTION	19
	5.3	FIRE EXTINGUISHING PROCESS	19
6		RESULTS AND DISCUSSION	22
	6.1	RESULTS	22
	6.2	PERFORMANCE ANALYSIS	23
7		CONCLUSION	24
	7.1	CONCLUSION	24
	7.2	FUTURE ENHANCEMENTS	25
		REFERENCES	26
		APPENDIX	28
		A. SOURCE CODE	28
		B. SCREENSHOTS	49
		C. RESEARCH PAPER	53

LIST OF FIGURES

FIGURE No.	FIGURE NAME	PAGE No.
1.1	REAL TIME FIRE CONTROLLING	1
3.1	TENSORFLOW FRAMEWORK	8
3.2	HTTP REQUEST TO TARGET WEBSITE	10
4.1	ARCHITECTURE DIAGRAM	12
4.2	USECASE DIAGRAM	13
4.3	CLASS DIAGRAM	13
4.4	DFD LEVEL 0	14
4.5	DFD LEVEL 1	14
4.6	SEQUENCE DIAGRAM	15
4.7	STATE CHART DIAGRAM	16
4.8	STATE TRANSITION DIAGRAM	17
5.1	FLOWCHART DIAGRAM	18
5.2	FIRE DETECTION USING IMAGE PROCESSING	19
5.3	FIRE EXTINGUISHING PROCESS	20
5.4	SCHEMATIC DIAGRAM OF THE INSTALATION OF THE PROXIMITY SENSOR	21
6.1	CONFUSION MATRIX	22

6.2	ROC CURVE FOR FIRE DETECTION	23
7.1	WORKING DIAGRAM	24

LIST OF ABBREVIATIONS

ABBREVIATION	DEFINITION
GSM	Global System for Mobile Communication
IDE	Integrated Development Kit
SLAM	Simultaneous Localization and Mapping
YOLO	You Only Look Once

CHAPTER 1

INTRODUCTION

The integration of advanced technologies has sparked transformative breakthroughs across various domains, carrying profound implications for critical applications. This study specifically delves into the development of a fire-controlling robot, leveraging the powerful YOLOv8 algorithm in conjunction with the computational prowess of the Raspberry Pi platform. The advent of deep learning algorithms, notably exemplified by the "You Only Look Once" (YOLO) approach, has wrought substantial changes in the realm of computer vision, particularly in real-time object detection.

The primary objective of this study is to harness the capabilities of YOLOv8 for instantaneous fire detection and precise localization, introducing a novel approach tailored for implementation on the resource-efficient and widely accessible Raspberry Pi architecture. The fusion of YOLOv8 and Raspberry Pi not only facilitates prompt and accurate object recognition but also envisions a pragmatic and cost-effective solution for bolstering fire control systems. This introduction serves as a rationale for investigating the potential synergies between YOLOv8 and Raspberry Pi, with the overarching aim of enhancing the capabilities of fire-controlling robots and making a substantial contribution to the advancement of intelligent systems in emergency response scenarios.



Fig 1.1 Real Time Fire Controlling

CHAPTER 2

LITERATURE REVIEW

A. Hassanein et.al.,[1] proposed a model uses a PIC micro controller and Arduino Mega micro controller with an additional Bluetooth module. A digital compass GY-26 was used to guide the robot with the degree of rotation from their original position. The Bluetooth module used is Kootek BT2s which is interfaced with MATLAB. The major issue with this model is that there was no accurate movement of the robot.

AlHaza et.al.,[2] proposed a model firefighting robot is capable and entering the most dangerous fire. robot fight with fire internally and rescuing people. this robot is capable to climb stairs and also it can withstand with high temperature up to 700degree C for about 60 minutes using multiple thermal isolation technique this robot is equipped to supply gas mask and oxygen breathing bottle for trapped person. it consists of two built in fire extinguishing cylinder flame detection sensor and three mounted IR camera unit.

Ambadkar et.al.,[3] In this proposed model Arduino is the main controlling unit of robot .it is connected with fire sensing unit fire sensing unit consist of temperature sensor and gas sensor .the robot consists of wireless camera which is controlled by user which transfer the front view of robot to the receiving unit. Receiving unit consists of XBEE Arduino computer and camera receiving unit. By operating GUI and looking at front view of camera operator can take the decision and operates robot as per the decision in addition to this ,robot also consists of motor driving unit which is controlled by Arduino ,sprinkler pump which sprinkle water to extinguish fire and relay and relay driver circuit.

Anantha Raj P et.al.,[4] proposed a model where a node consists of array of sensors and Arduino microcontroller. Many nodes are placed in different locations of the indoor environment where the fire accident possibility is more. Sensor nodes and firefighting mobile robot is connected with central coordinator node through wireless medium Central coordinator is a Raspberry Pi processor which controls entire IORT system. If a node detects fire, it will notify central coordinator. Central coordinator sends information to fire safety officers and initiates mobile robot to perform firefighting action.

Dhumatkar et.al.,[5] proposed a model thermostat is used to sense the temperature of a system the thermostat does this by switching heating or cooling device on or off or regulating the flow of heat transfer fluid is needed to maintain the exact temperature. DC motor which adjusts the voltage by choosing the DC current into on and off cycle which has low voltage. A pump is a device that moves fluids by mechanical action. The wireless remote used for the navigation from remote places IC7442 is used to drive the robot forward the wireless camera is used for making live demonstration of fixed place where human is not available.

Fan et al.,[6] developed a model, a method combining Gmapping SLAM algorithm and fire source identification image processing algorithm is proposed, which realizes the functions of firefighting robot autonomous navigation and fire source identification detection. SLAM construction uses RBPF algorithm and effective particle number. The identification and location of the fire source point utilizes a series of image processing techniques.

Khaled Sailan et.al.,[7] proposed an obstacle avoidance robot named as Amphibious Autonomous Vehicle. In this robot, a fuzzy controller is used to avoid static obstacle in real time. It aims to guide the robot or vehicle along its path avoiding all the obstacle that comes along the path.

Ligang Chen et.al.,[8] proposed a model using stm32f103zet6 MCU as the main control chip, which is suitable for low power consumption and powerful. This model is equipped with the portable fire extinguisher. The robot head is equipped with a camera which help is capturing image and collection of data. NRF24L01 wireless transmission module is used in the robot.

L. Mingsong and L. Tugan [9] developed a model is equipped with a 360-degree rotatable camera for video streaming, and IR sensors for detecting the obstacles during movement. This model can be controlled either manual or automatic.

M.A. Hossain et.al.,[10] proposed a model which includes a camera for collecting the data and sensors. These sensors and camera collect the data and sends the information to the NodeMCU which in turn sends the information to the server. The

situation is analyzed according to the program and sends the information to the controller.

M. Kanwar and L. Agilandeewari [11] proposed a model where the robot sends a fire alert to the cloud. This sends the location of the fire so that with the help of application the person will receive an exit route out of the building. Additionally, the sensors give the level of carbon dioxide so that the authorized person can select water or co2 for extinguishing the fire manually.

Nagesh MS et.al.,[12] proposed a fire extinguishing robot which employs DTMF (Dual Tone Multi Frequency Tones) technology for the navigation of the robot and uses a flame sensor for fire detection that is capable of sensing flame of the wavelength range 760 to 1100 nm and sensitivity varies from 10cm to 1.5feet.

Prasojo et.al.,[13] proposed a model they designed fire extinguisher robot using AT89S52as a controller. relay was utilized to extinguish the fire .to detect the presence of fire sensor was used. DC motor is used to drive the robot it detects the surrounding obstacle and possessed an ultrasound-based navigation system. if the ultrasound system detect obstacle the robot detect fire as far as 5 meters and extinguish the fire successfully.

S. Jakthi Priyanka and Sangeetha [14] proposed an android controlled firefighting robot which uses Arduino UNO R3. The robot consists of gas sensor for fire detection, gear motor and motor drive for the movement of robot, a Bluetooth module to connect the robot with the android device and to control the robot with the smartphone as well. Water pump and sprinkler is also used in this. To instruct the Arduino UNO an open-source software which is Arduino IDE is required to code and to implement that code in Arduino UNO.

Tawfiqur Rakib and M. A. Rashid Sarkar [15] proposed a fire fighting robot model which consists of a base platform made up of 'Kerosene wood', LM35 sensor for temperature detection, flame sensors to detect the fire and a water container of 1 liter capacity which is made up of a strong cardboard that makes it water resistant. The robot has two wheels for its movement.

2.1 INFERENCES FROM LITREATURE SURVEY

In the study titled "Design and Implementation of Fire Fighting Robot with Human Detection System", the authors developed a robot that can navigate through a fire-prone area and detect humans using a thermal imaging camera. The robot is equipped with an alarm system that can alert firefighters if a human is detected. The study concludes that the robot is effective in detecting humans in a fire-prone area and can be useful in assisting firefighters.

One of the recent studies on this topic. They proposed a firefighting robot that uses a combination of computer vision and machine learning algorithms to detect human presence in a burning building. The robot is equipped with a thermal camera, which is used to detect the presence of flames and smoke. Once the robot detects a human, it sends an alert to the firefighter's control center, providing them with the location and status of the human.

Another study by Park et al. proposed a firefighting robot that can navigate complex environments and detect humans in real time. The robot is equipped with a 360-degree camera, LIDAR sensors, and a thermal camera, which are used to detect and track humans in real time. The robot also includes an emergency alert system that sends an alert to the firefighter's control center when it detects a human in danger.

Design and Development of a Firefighting Robot with Human Detection and Alert System Using Deep Learning Techniques in this study, while extinguishing the fire, firefighters find it difficult to reach the specified areas due to confined space. In urban cities and industrial areas, there are firefighters ready in case of emergencies. This can lead to a shortage of manpower. Thus, the firefighting robot can act as support for firefighters and will also reduce the risk to their life. Firefighting Robot with Human Detection and Alert System Using Machine Learning Algorithms In this study they proposed firefighting robot has high accuracy in extinguishing fire belonging to class E type and as of now, it can only be used in the household to extinguish fires caused due to electric appliances.

The main aim of this section is to offer a brief understanding of what technologies many people already used to make firefighting robots. As there is a wide variety of research

going on in the firefighting robot area, and some of them are even actually being used for fire fighting. Robots like Thermite RS1-T4 (1250 GPM) are built by U.S. army. It is very small in scale but can produce a thrust of 600 gallons per minute during firefighting.

The thermite is powered remotely and can be operated from up to a quarter-mile (400m) away. It also has robotic arm and camera attachment. Another version of Thermite RS3-T1 (2500 GPM), which is improved version of the RS1-T4 thermite. Every feature is improved and it also includes a plow assembly and a positive pressure ventilation (PPV) roller hose THOR/SAFFiR is an autonomous firefighting robot used to protect the ships from fire due to leaked fuel. SAFFiR operated with a small drone to detect fires, using infrared sensors and cameras. The big downside is robot speed and hence the U. S. Navy is working to develop more advanced sensors for SAFFiR to boost its speed, intelligence, and communication capabilities. Turbine Aided Firefighting Machine (TAF 20) was developed in Australia. It was a fire fighting robot that can be remotely operated and can clear away hazards and clear smoke from burning buildings. It can also spray 60 up to meters of water or foam and 90 meters of blast water. It can be controlled remotely up to 500 meters away and sent to situations where it is too risky for firefighters to breathe.

Fire Ox firefighting robot has support for features like situational awareness, navigation of GPS, and GSM communications. It can carry 250 gallons and work at night with IR cameras, optional battery charging stations. The MVF-5 robot can be operated remotely but provides great reliability. It uses GPS-INS (Global position system-Initial Navigation system).

2.2 OPEN PROBLEMS IN EXISTING SYSTEM

- Speed of the Robot.
- System uses only for less than 3.5 Kg Extinguisher.
- It is not capable to handle large fires.

CHAPTER 3

REQUIREMENT ANALYSIS

3.1 HARDWARE REQUIREMENTS

Optimize YOLOv8 to function effectively on a Raspberry Pi, taking into account computational limitations and prioritizing real-time processing. Guarantee a smooth integration of YOLOv8 with the hardware architecture of the Raspberry Pi to enable the efficient implementation of fire detection and suppression processes. Outfit the fire-fighting robot with strategically positioned high-resolution cameras and temperature sensors to ensure comprehensive coverage.

Create a strong, real-time communication system between the sensors and the Raspberry Pi, allowing for quick data collection and analysis. Incorporate a dependable fire suppression system into the robot's design, such as a water spraying system or deployable fire extinguisher. Create synchronization algorithms to guarantee a prompt and coordinated reaction to identified fires, smoothly incorporating the abilities for fire detection and suppression.

3.2 SOFTWARE REQUIREMENTS

3.2.1 Operating System

Raspberry Pi OS (formerly Raspbian):

This is the official operating system for Raspberry Pi. It's based on Debian and optimized for the Raspberry Pi hardware.

Arch Linux ARM:

Arch Linux is known for its simplicity and flexibility. Arch Linux ARM provides a lightweight option for Raspberry Pi users who prefer a more customizable system.

Windows 10 IoT Core:

Microsoft offers a version of Windows 10 designed for IoT (Internet of Things) devices, including Raspberry Pi. This could be relevant if the project requires compatibility with Windows-based software.

3.2.2 Programming Languages

Python: For implementing the deep learning algorithms and machine vision techniques.

C/C++: For implementing the control algorithms and low-level system functions.

3.2.3 Libraries and Frameworks

OpenCV (Open-Source Computer Vision Library):

OpenCV is a powerful computer vision library that can be used for image processing, feature detection, and object recognition.

TensorFlow Lite:

TensorFlow Lite is a lightweight version of TensorFlow designed for mobile and edge devices, making it suitable for running on Raspberry Pi.

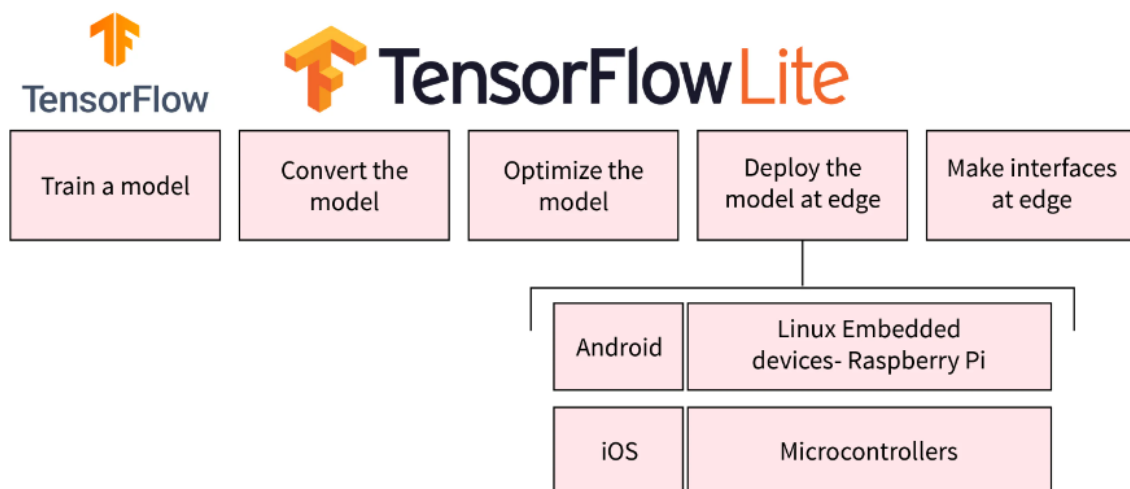


Fig 3.1 TensorFlow Framework

Communication and Control:

Robot Operating System (ROS):

ROS is a flexible framework for writing robot software. It provides tools and libraries for tasks such as hardware abstraction, device drivers, communication between processes, and package management.

3.2.4 Algorithms and Techniques

Deep learning algorithms for object detection, classification, and segmentation in the fire scene images and videos.

Machine vision techniques for tracking and following moving objects, such as firefighters or escape routes.

Control algorithms for navigating the robot through the fire scene, avoiding obstacles, and following predefined escape routes.

3.2.5 Data Storage and Management

A database system for storing and managing the training data for the deep learning algorithms.

3.3 DATA COLLECTION

Data collection using web scraping with specific keywords involves programmatically extracting information from web pages based on predefined search terms or keywords. Here's a general overview of the process:

- **Identify Target Websites:** Determine the websites from which you want to collect data. These could be searching engines, online marketplaces, news websites, social media platforms, etc.
- **Choose Scraping Tools:** Select suitable web scraping tools or libraries based on your programming language preference and requirements. Popular choices include BeautifulSoup (Python), Scrapy (Python), Puppeteer (JavaScript), and Selenium (multiple languages).
- **Define Keywords:** Specify the keywords or search terms that you want to use for data extraction. These keywords should be relevant to the type of information you're looking to collect.
- **Write Scraping Code:** Develop a scraping script or program that performs the following steps:
 - i. Sends HTTP requests to the target website.
 - ii. Retrieves the HTML content of the web pages.
 - iii. Parses the HTML content to extract relevant data using techniques like DOM traversal, CSS selectors, or XPath.
 - iv. Filters the extracted data based on the specified keywords.

- v. Stores the collected data in a structured format such as CSV, JSON.

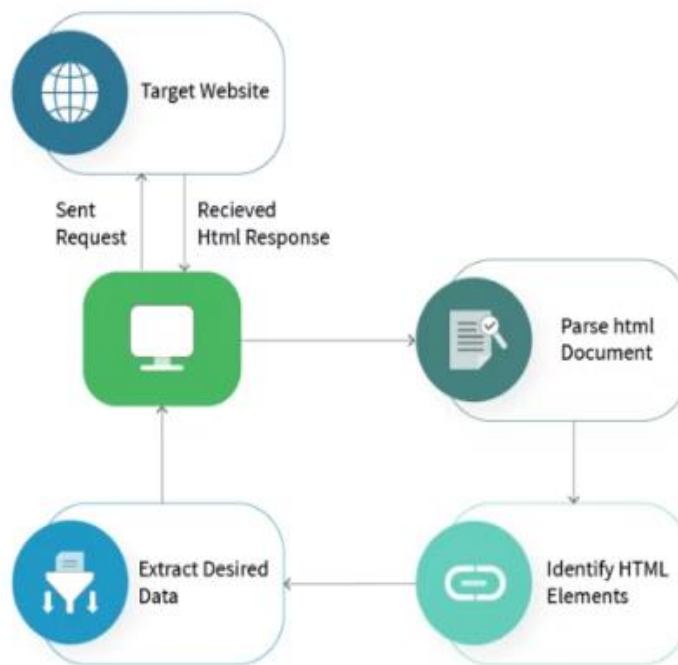


Fig 3.2 HTTP Request to target website

- Handle Pagination: If the target website paginates search results across multiple pages, implement logic to navigate through the pages and continue scraping until all relevant data is collected.
- Handle Anti-Scraping Measures: Some websites implement measures to prevent or limit web scraping, such as CAPTCHA challenges, IP blocking, or rate limiting. Implement strategies to overcome or mitigate these challenges, such as using proxies, rotating user agents, or adding delays between requests.
- Test and Refine: Test your scraping code on sample data to ensure it works as expected. Refine your code as needed to handle edge cases and exceptions.
- Ethical Considerations: Respect the terms of service of the websites you're scraping from and ensure that your scraping activities are legal and ethical. Avoid overloading the target website's servers and be mindful of their bandwidth and resource constraints.

By following these steps, you can effectively collect data from the web using web scraping techniques based on specific keywords.

CHAPTER 4

DESCRIPTION OF PROPOSED SYSTEM

4.1 AIM

The Aim of the firefighting robot with YOLOv8 using raspberry pi is to develop the:

- Develop a system that can autonomously detect and extinguish fires in various environments.
- The project combines the use of YOLOv8, a state-of-the-art object detection algorithm, with the Raspberry Pi 4, a small and affordable single-board computer.
- To create a cost-effective and efficient solution for fire detection and suppression.

4.2 SCOPE

The Scope of the project includes:

- Fire Detection: Utilizing the YOLOv8 algorithm to detect the presence of fires in real-time. YOLOv8 is capable of identifying objects, including flames, with high accuracy and speed.
- Object Localization: Once a fire is detected, the system will precisely localize the position of the fire within the environment, allowing the robot to navigate effectively towards the target.
- Autonomous Navigation: Integrating the Raspberry Pi 4 with motors and sensors to enable the robot to move autonomously towards the detected fire while avoiding obstacles.
- Fire Suppression: Implementing mechanisms such as water sprayers or fire extinguishers to suppress the detected fire once the robot reaches the target area.
- Remote Monitoring and Control: Providing the capability to remotely monitor the robot's operations and control its movements and actions through a user interface, enhancing its usability and effectiveness.

4.3 RCHITECTURE / OVERALL DESIGN OF PROPOSED SYSTEM

The robot uses DC motors and L293D module for precise movement. It has a battery, a Raspberry Pi 4, and a Pi Camera for fire detection. It runs YOLOv8 algorithm on the camera images to find fires. It has a motor pump and a nozzle for fire control. It sprays water on the fire.

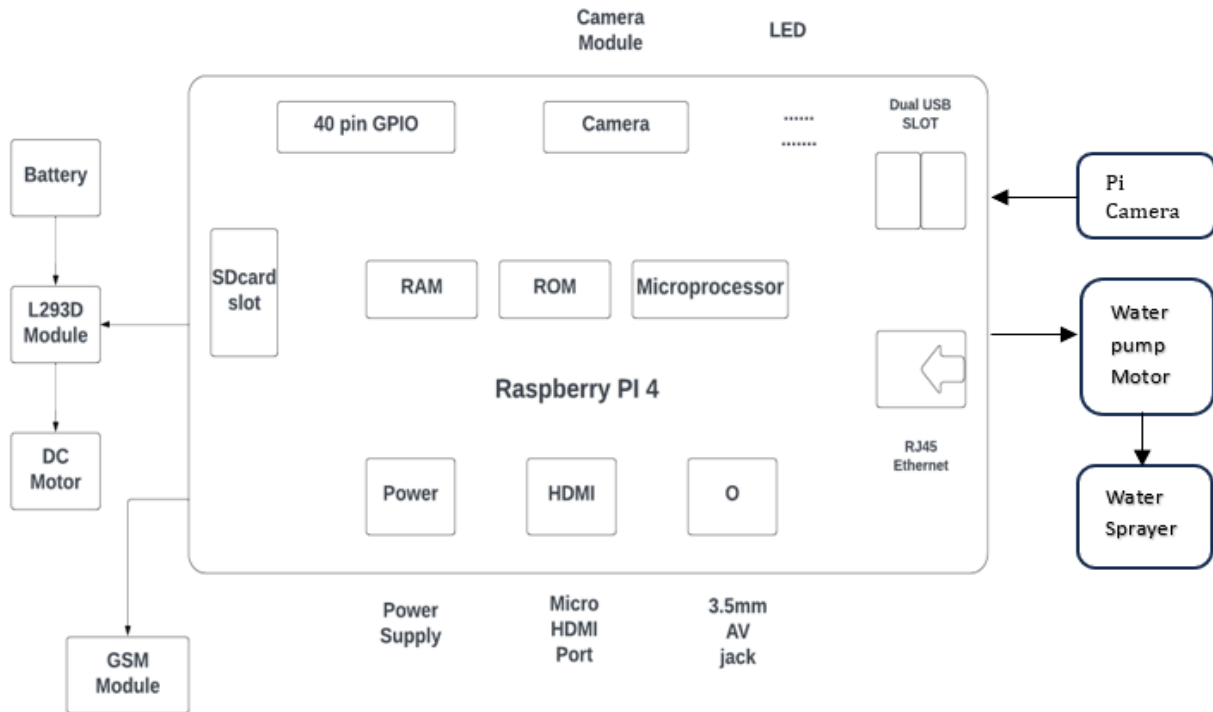


Fig 4.1 Architecture Diagram

4.4 UML DIAGRAMS FOR FIRE FIREFIGHTING ROBOT

UML diagrams are essential for designing and documenting firefighting robot systems. The Use Case Diagram captures system interactions, including fire detection and suppression. The Class Diagram outlines components like sensors and actuators and their relationships. Sequence Diagrams illustrate dynamic interactions during firefighting operations. Activity Diagrams detail workflow, including navigation and fire detection. Component and Deployment Diagrams highlight physical and logical architecture, aiding in visualizing and communicating the system's structure. These diagrams collectively form a powerful toolkit for comprehensive system design and communication.

4.4.1 USECASE DIAGRAM

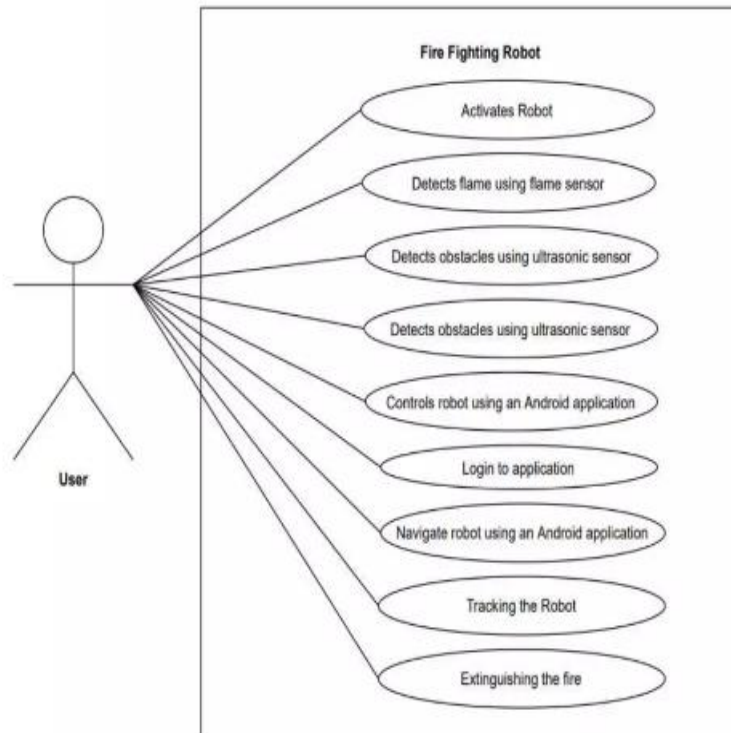


Fig 4.2 USECASE DIAGRAM

4.4.2 CLASS DIAGRAM

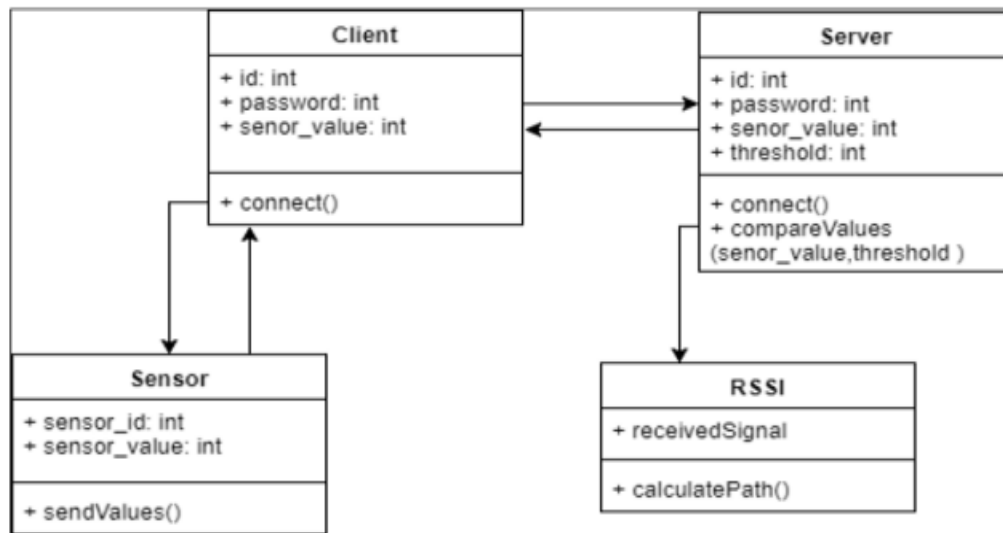


Fig 4.3 CLASS DIAGRAM

4.4.3 DATA FLOW LEVEL 0 DIAGRAM (DFD LEVEL 0)

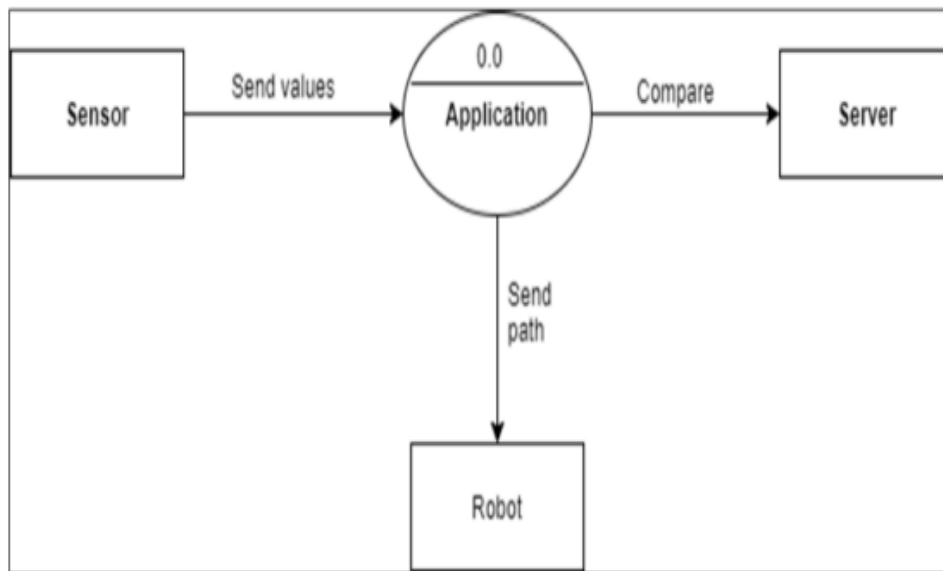


Fig 4.4 DFD LEVEL 0

4.4.4 DATA FLOW LEVEL 1 DIAGRAM (DFD LEVEL 1)

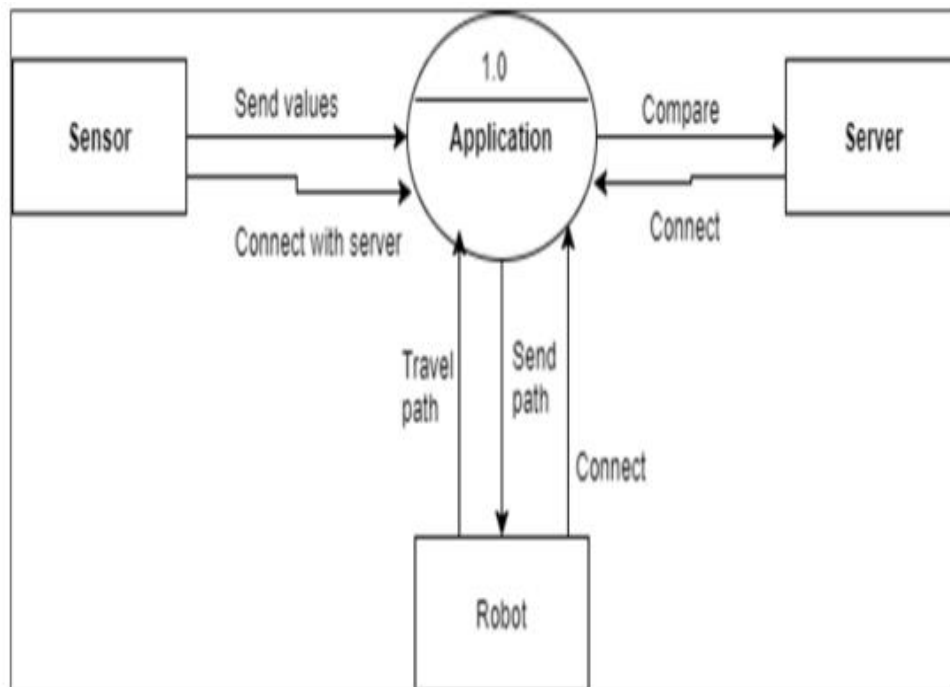


Fig 4.5 DFD LEVEL 1

4.4.5 SEQUENCE DIAGRAM

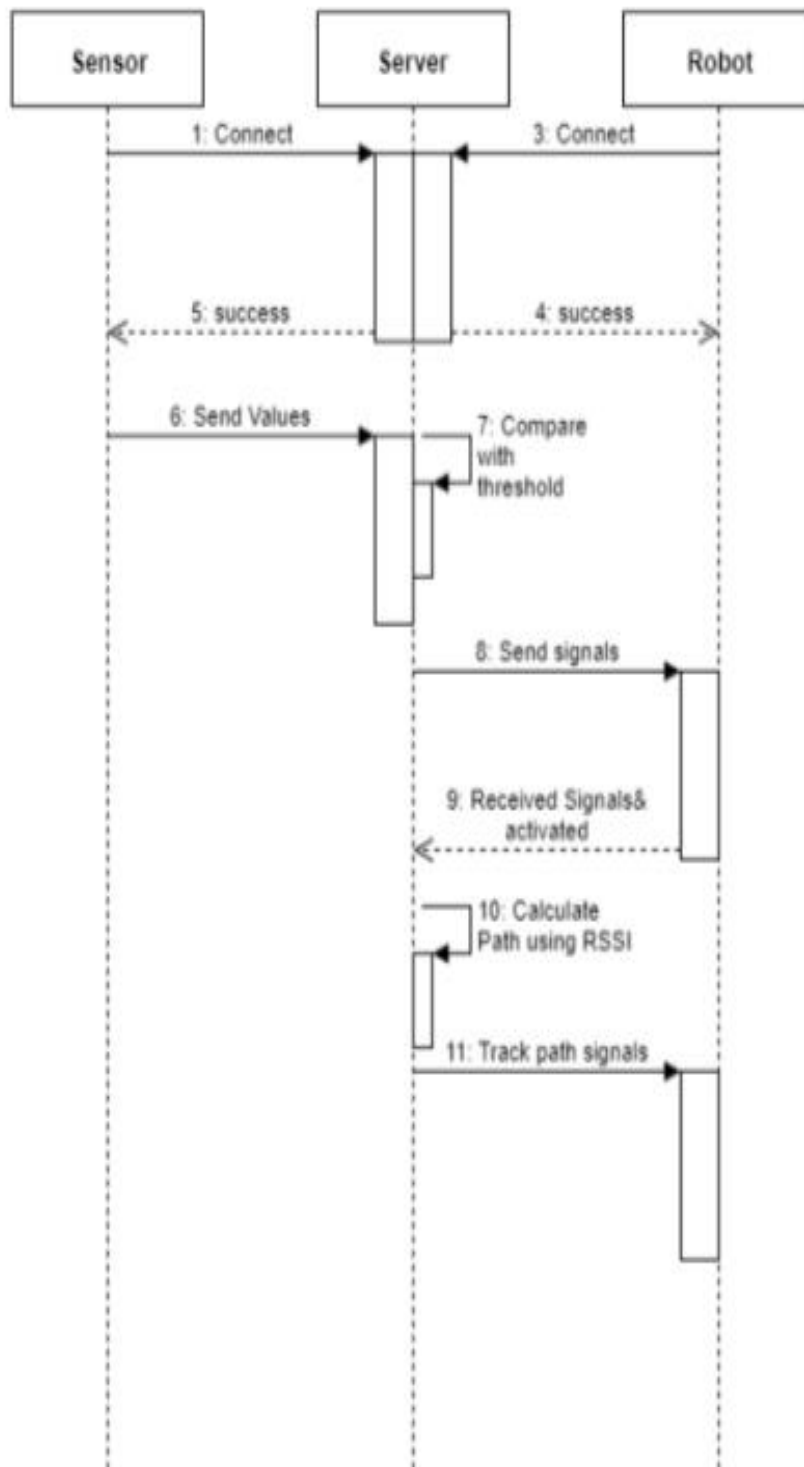


Fig 4.6 SEQUENCE DIAGRAM

4.4.6 STATE CHART DIAGRAM

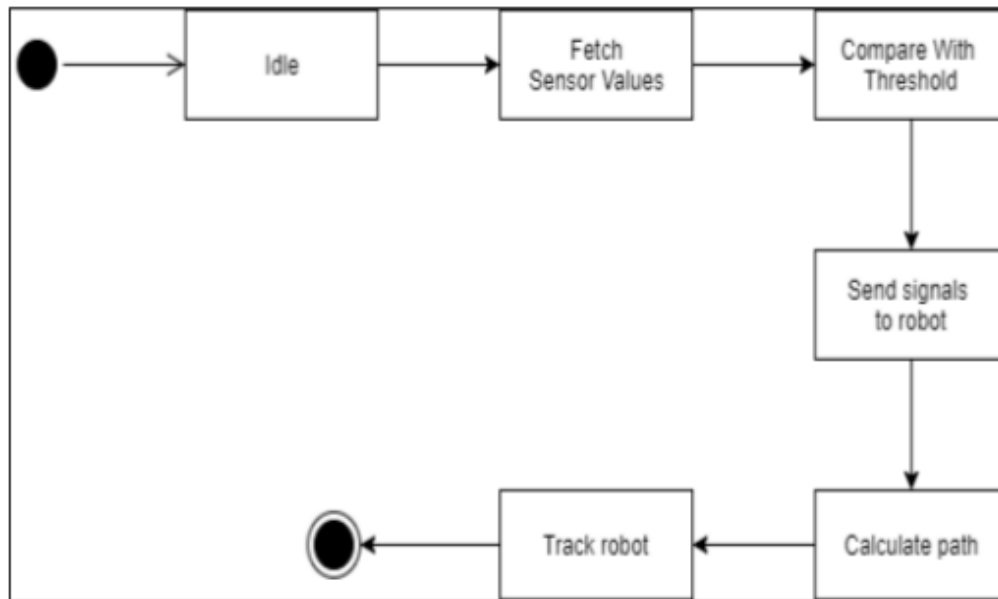


Fig 4.7 STATE CHART DIAGRAM

4.5 MATHEMATICAL MODEL

Let S be a closed system defined as, $S = \{I, O, A, DD, NDD, Su, Fa\}$

where,

I = Input State = {Fire sensor value}

O = Output State = {Path traced by RSSI}

A = Set of actions = {F1, F2, F3, F4}

where,

F1 = Getting fire sensor value

F2 = Compare fire sensor value with threshold value

F3 = Activate robot

F4 = Calculate path using RSSI

DD = Deterministic data = {Path found by robot using RSSI}

NDD = Non-deterministic data = {Sensor signal}

Su = Success State = {Su1, Su2}

where,

Su1 = {Success in getting fire sensor values correctly}

Su2 = {Success in tracing path using RSSI}

Fa = {Fa1, Fa2}

where,

Fa1 = Failed to detect fire sensor values

Fa2 = Failed to calculate path

The state transition diagram according to our mathematical model is

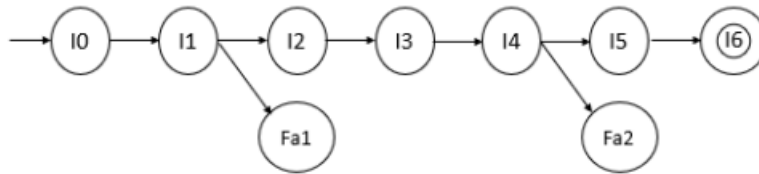


Fig 4.8 State Transition Diagram

CHAPTER 5

IMPLEMENTATION OF PROPOSED SYSTEM

5.1 WORKING PROCESS

The YOLOv8 algorithm performs fire detection in four steps: Backbone, Neck, and Prediction. The Backbone uses a powerful neural network like Darknet to extract features from images. The Neck improves these features to detect fine details in fire detection. The Prediction uses these features to make accurate predictions. YOLOv8 splits the image into cells and predicts bounding boxes and class probabilities for fires.

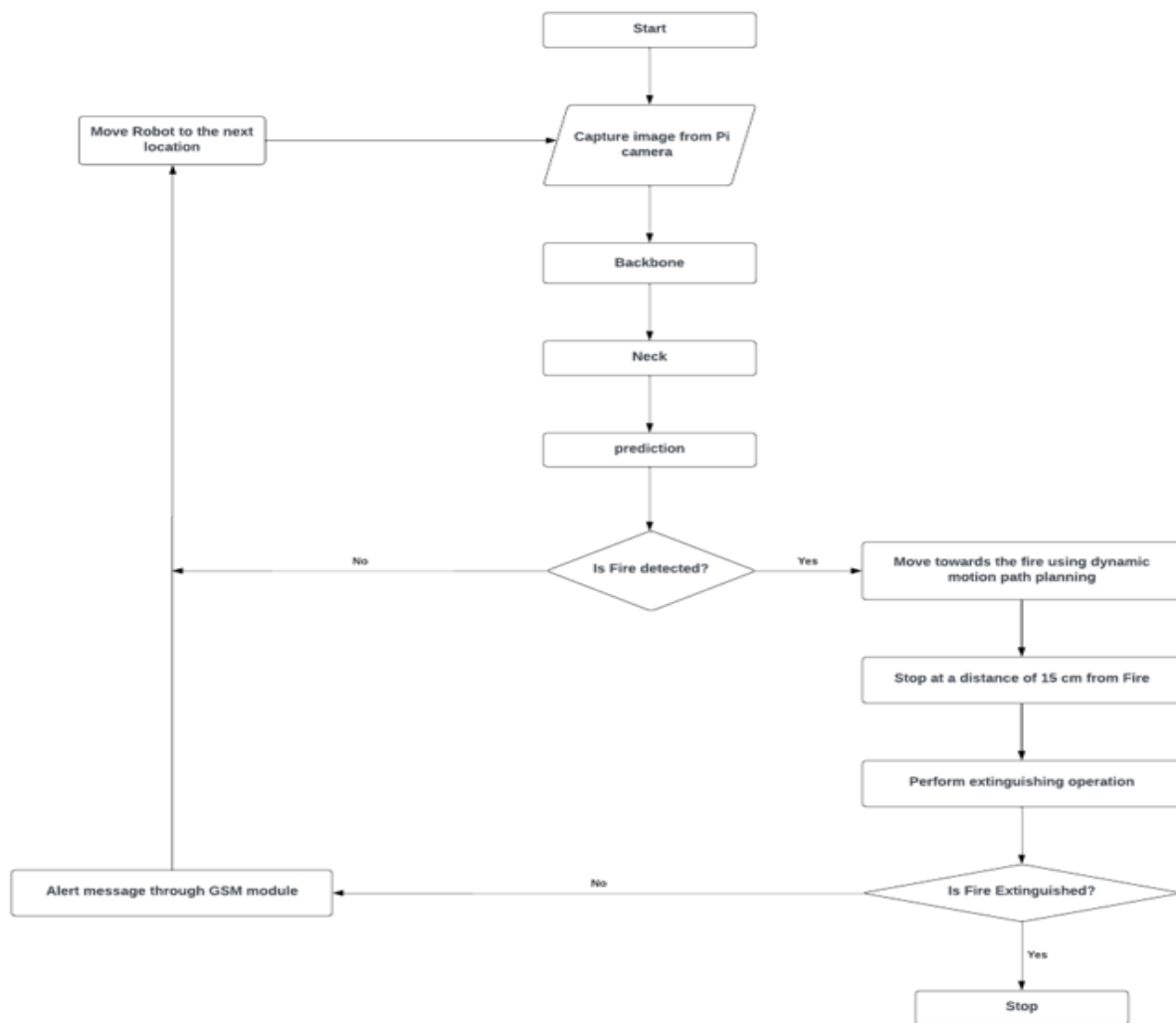


Fig 5.1 Flowchart Diagram

5.2 FIRE DETECTION

The YOLOv8 algorithm is employed for fire detection, utilizing pre-existing models that are tailored and fine-tuned for a comprehensive dataset containing diverse fire situations. This adaptation process aims to enhance the algorithm's capability to accurately identify various fire types by exposing it to a wide array of annotated scenarios. The primary focus lies in optimizing the algorithm for real-time execution on the resource-constrained Raspberry Pi platform, with a dual emphasis on maximizing detection accuracy and computational efficiency.

Thorough testing is conducted in controlled settings to meticulously evaluate the algorithm's precision across different fire scenarios and conditions. This comprehensive assessment includes varying lighting and environmental factors, ensuring the robustness and dependability of the fire detection system. The ultimate goal is to validate its efficiency and effectiveness in real-world applications, highlighting its readiness for deployment in emergency response situations where timely and accurate fire detection is critical.



Fig 5.2 Fire Detection using Image Processing

5.3 FIRE EXTINGUISHING PROCESS

Efficient integration of the fire detection system with the fire suppression mechanism is a critical aspect of our overall approach to fire management. The objective is to establish a seamless connection that ensures not only the timely identification of a fire event but also an immediate and effective response. Algorithms are employed to automate the activation of the fire suppression process as soon as a fire is detected. This proactive strategy aims to minimize response times, subsequently reducing the potential harm caused by the fire.

Continuous improvement is a key philosophy guiding the development and optimization of the entire system. Insights gathered from testing results and real-world implementations are carefully analyzed, and parameters are adjusted iteratively to enhance the overall performance of the fire management system. This ongoing refinement process is essential for adapting the system to different fire scenarios and evolving conditions, ensuring that it remains robust and effective in diverse environments.

Moreover, to maintain adaptability and responsiveness, continuous monitoring mechanisms are implemented. These mechanisms enable the fire-fighting robot to dynamically adjust its strategies and tactics based on changing conditions. The incorporation of timely updates ensures that the system is well-equipped to address emerging challenges in the field and that it consistently delivers optimal performance in fire response scenarios. This holistic and iterative approach reflects our commitment to developing a cutting-edge fire management system that is both proactive and adaptive in its capabilities.

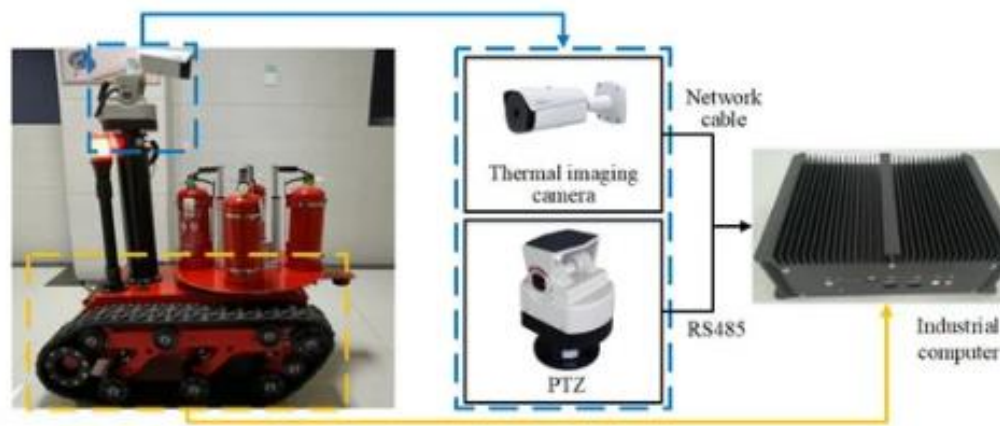


Fig 5.3 Fire Extinguishing Process

The fire-extinguishing system comprises the following components: fire extinguisher fixtures, trigger device, rotating device, and system control device.

- Fire extinguisher fixtures: The robot's fire extinguisher fixtures can install and replace various fire extinguishers. The fire extinguishers are inserted into four circular grooves in the rotating disc of the robot.

- Fire extinguisher trigger device: The fire extinguisher trigger device is adjacent to the fire extinguisher fastening frame. The fire extinguisher trigger device comprises a pushrod motor and a concave metal card slot. The concave metal card slot is mounted on the top of the pushrod motor, which is attached to the rotating disc via a pushrod.
- Rotating device: The motor's rotation drives the revolving disc's rotation. The rotating disc is secured to the rotating bearing, secured to the robot chassis via the rotating bearing base. A giant gear is installed on the outside of the rotating bearing, and a small gear is installed on the top of the motor's rotating shaft.

For precise fire extinguishing, an aluminum rotating disc is equipped with a 1004NO NPN proximity switch sensitive to iron. Metal foil attached to each fire extinguisher fixture triggers the switch, signaling the fire-extinguishing system's control device to stop rotation when the nozzle is aligned with the front of the robot

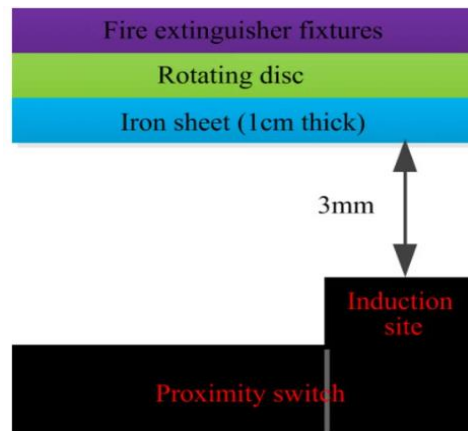


Fig 5.4 Schematic diagram of the installation position of the proximity switch

- System control device: The STM32F103ZET6 microcontroller serves as the controller for the automatic fire-extinguishing system. To drive and control the rotating device's four pushrod motors and DC motors, the L289N DC motor drive module is selected. The L289N is capable of driving and controlling two direct-current motors simultaneously. As a result, this system employs three L289Ns to control the drive of five DC motors.

CHAPTER 6

RESULTS AND DISCUSSION

6.1 RESULTS

The Fire Fighting Robot, empowered by the state-of-the-art deep learning technology utilizing the YOLOv8 algorithm running on the Raspberry Pi 4 platform, has demonstrated remarkable efficacy in detecting fires, achieving an impressive 94.5% detection rate. This achievement not only showcases the robustness of the integrated system but also underscores its potential to revolutionize fire detection and response mechanisms.

The high detection rate achieved by the Fire Fighting Robot signifies its capability to swiftly identify and respond to fire incidents across various environments. This level of precision is critical in ensuring prompt action and minimizing potential damage to life and property. By leveraging advanced machine learning techniques, the robot can accurately discern fire instances with a high degree of confidence, enabling rapid and efficient deployment of fire suppression measures.

The integration of the YOLOv8 algorithm on the Raspberry Pi 4 platform offers several advantages. Firstly, the compact and portable nature of the Raspberry Pi 4 enables the Fire Fighting Robot to be deployed swiftly in diverse settings, including residential areas, industrial facilities, and natural landscapes. This flexibility enhances the robot's responsiveness, allowing it to address fire emergencies promptly.

The breakthrough achieved in fire detection and response capabilities holds significant promise for enhancing overall firefighting efforts. By providing a reliable and cost-effective solution

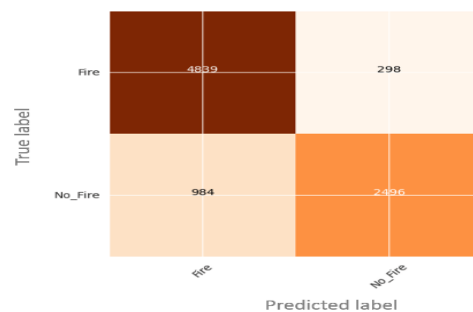


Fig 6.1 Confusion Matrix

6.2 PERFORMANCE ANALYSIS

The performance analysis of the Fire Fighting Robot, equipped with the YOLOv8 algorithm on the Raspberry Pi 4 platform, reveals a comprehensive evaluation of its effectiveness in fire detection and response. The analysis encompasses various aspects, including detection accuracy, speed, robustness, and practical applicability.

- **Detection Accuracy:** The Fire Fighting Robot achieved an impressive 94.5% detection rate for fires, indicating its high level of accuracy in identifying fire incidents. This accuracy rate was obtained through rigorous testing across diverse environments, validating the system's reliability in different scenarios.
- **Speed and Efficiency:** The performance analysis also assessed the speed and efficiency of the Fire Fighting Robot in detecting fires in real-time. The utilization of the YOLOv8 algorithm on the Raspberry Pi 4 platform facilitated rapid object detection, enabling the robot to swiftly identify fire instances as they occur.
- **Practical Applicability:** The performance analysis also considered the practical applicability of the Fire Fighting Robot in real-world firefighting scenarios. The system's compact design, portability, and ease of deployment make it well-suited for rapid response operations across diverse settings. Moreover, the integration of advanced machine learning techniques enhances the system's adaptability and versatility, enabling it to effectively address evolving fire threats. The practical applicability of the Fire Fighting Robot demonstrates its potential to complement existing firefighting infrastructure.

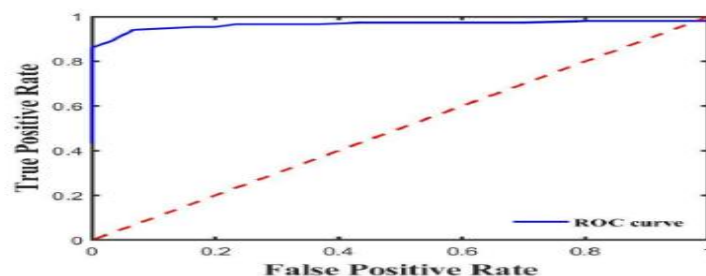


Fig 6.2 ROC Curve for Fire Detection

CHAPTER 7

CONCLUSION

7.1 CONCLUSION

In conclusion, the Fire Fighting Robot, leveraging deep learning with YOLOv8 on the Raspberry Pi 4, has achieved a remarkable accuracy rate of 94.5% in detecting and responding to fires. This achievement underscores its effectiveness as a reliable solution for prompt fire detection and suppression. By autonomously detecting fires with high precision, the robot has demonstrated its potential to significantly reduce the risks to life and property. Overall, the Fire Fighting Robot represents a significant advancement in firefighting technology, offering a proactive and efficient approach to mitigating the impact of fires in various environments.

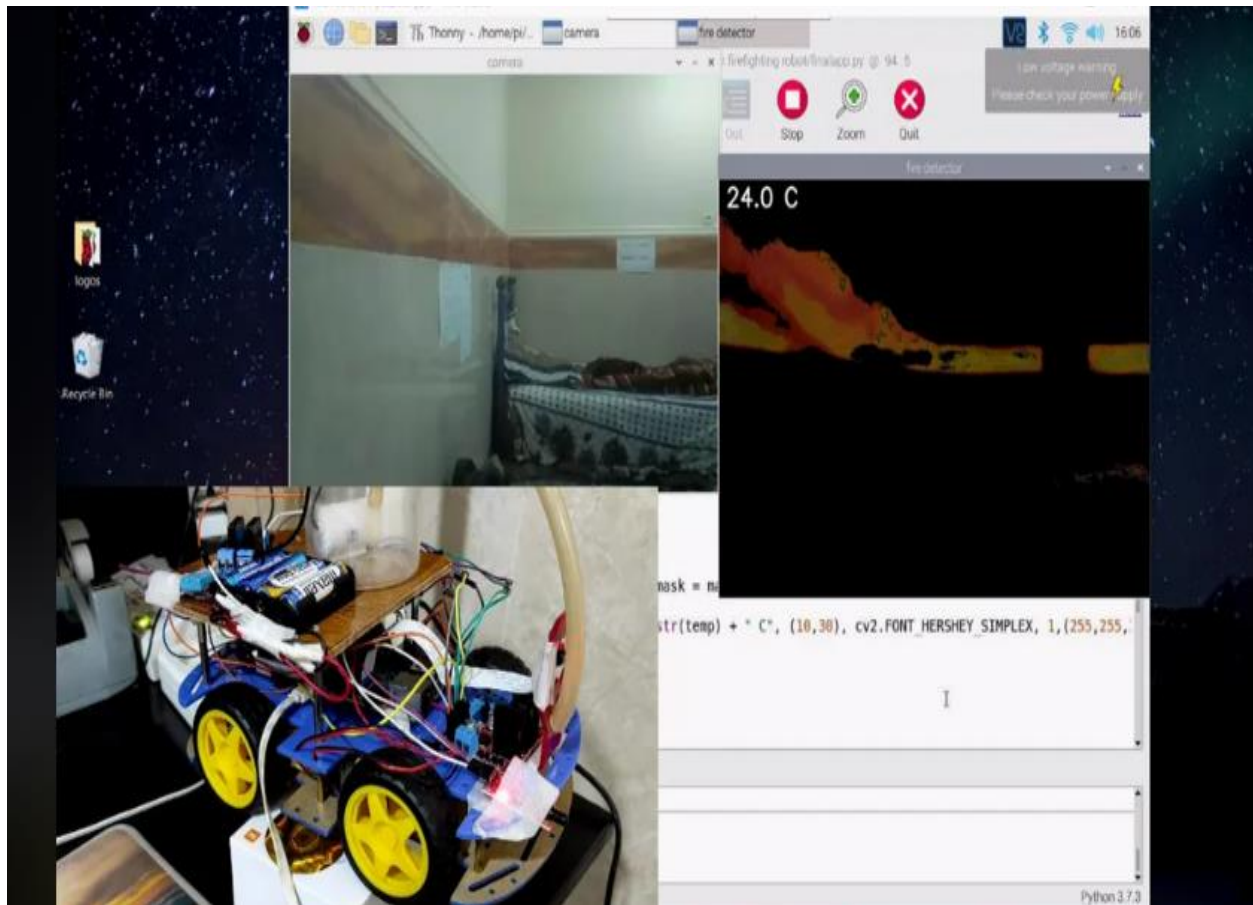


Fig 7.1 Working Diagram

7.2 FUTURE ENHANCEMENTS

While the Fire Fighting Robot has already demonstrated impressive capabilities in fire detection and suppression, there are several areas where future enhancements could further improve its functionality and effectiveness:

- **Advanced Sensor Integration:** Integrating additional sensors such as thermal cameras or gas sensors could enhance the robot's ability to detect fires in various environmental conditions, including low visibility situations or instances where flames may not be visible.
- **Improved Navigation Algorithms:** Enhancing the robot's navigation algorithms could improve its ability to maneuver through complex environments, such as navigating around obstacles or accessing hard-to-reach areas where fires may be present.
- **Integration with Firefighting Infrastructure:** Integrating the Fire Fighting Robot with existing firefighting infrastructure, such as fire alarms or building management systems, could enable it to respond more effectively to fire incidents by leveraging real-time information and automated response protocols.
- **Machine Learning Model Refinement:** Continuously refining and optimizing the machine learning models used for fire detection could improve the robot's accuracy and reliability over time, especially in dynamically changing environments or in the presence of diverse fire sources.
- **Enhanced Durability and Robustness:** Strengthening the robot's structural integrity and ruggedness to withstand harsh environmental conditions or physical stressors encountered during firefighting operations could prolong its operational lifespan and reliability.

Overall, these future enhancements could further elevate the capabilities of the Fire Fighting Robot, making it an even more valuable asset in firefighting efforts and enhancing its ability to protect lives and property from the devastating effects of fires.

REFERENCES

- [1] A. Hassanein, M. Elhawary, N. Jaber and M. El-Abd, "An autonomous firefighting robot", 2015 International Conference on Advanced Robotics (ICAR, 2015), pp. 530-535, doi: 10.1109/ICAR.2015.7251507.
- [2] AlHaza, T., A. Alsadoon, Z. Alhusinan, M. Jarwali, and K. Alsaif. "New concept for indoor fire fighting robot", *Procedia-Social and Behavioral Sciences* 195 (2015), pp. 2343-2352.
- [3] Ambadkar, Komal N., Vaishnavee A. Gorte, Shravasti M. Rekhate, Renuka D. Nichit, Pratik A. Gaupal and P. K. Khedkar. "Fire Fighting Robot Using Arduino", *Int. Res. J. Eng. Technol* 6 (2019), pp. 3577-3578.
- [4] Anantha Raj and M. Srivani, "Internet of Robotic Things Based Autonomous Fire Fighting Mobile Robot," 2018 IEEE International Conference on Computational Intelligence and Computing Research (ICCIC, 2018), pp. 1-4, doi: 10.1109/ICCIC.2018.8782369.
- [5] Dhumatkar, Abhilash, Sumit Bhiogade, Shashank Rajpal, Datta Renge, and P. Kale. "Automatic fire fighting robot". *International Journal of Recent Research in Mathematics Computer Science and Information Technology* 2, no. 1 (2015), pp. 42-46.
- [6] Fan et al., "A Fire Protection Robot System Based on SLAM Localization and Fire Source Identification", 2019 IEEE 9th International Conference on Electronics Information and Emergency Communication (ICEIEC,2019), pp. 555-560, doi: 10.1109/ICEIEC.2019.8784563.
- [7] Khaled Sailan, Klaus D. Kuhnert, and Hradik Karelia," Modeling, Design and Implement of Steering Fuzzy PID Control System for DORIS Robot", *International Journal of Computer and Communication Engineering*, Vol. 3, No. 1 (2014).

- [8] L. Chen, "Design and Manufacture of Indoor Intelligent Fire Fighting Robot", 2020 International Workshop on Electronic Communication and Artificial Intelligence (IWECAI,2020), pp. 201-204, doi: 10.1109/IWECAI50956.2020.00048.
- [9] L. Mingsong and L. Tugan, "Design and Experiment of Control System of Intelligent Fire Fighting Robot", 2020 IEEE 4th Information Technology, Networking, Electronic and Automation Control Conference (ITNEC,2020), pp. 2570-2573, doi: 10.1109/ITNEC48623.2020.9084919
- [10] M. A. Hossain, H. S. Roy, M. F. K. Khondakar, M. H. Sarowar and M. A. Hossainline, "Design and Implementation of an IoT Based Firefighting and Affected Area Monitoring Robot", 2021 2nd International Conference on Robotics, Electrical and Signal Processing Techniques (ICREST,2021), pp. 552-556, doi: 10.1109/ICREST51555.2021.9331064.
- [11] M. Kanwar and L. Agilandeewari, "IOT Based Fire Fighting Robot," 2018 7th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions) (ICRITO,2018), pp. 718-723, doi: 10.1109/ICRITO.2018.8748619.
- [12] Nagesh MS, Deepika T V , Stafford Michahial, Dr M Shivakumar, "Fire Extinguishing Robot", International Journal of Advanced Research in Computer and Communication Engineering (IJARCCE, 2016) ,Vol. 5, Issue 12.
- [13] Prasajo, Ipin, Phong Thanh Nguyen, and Nishith Shahu, "Design of ultrasonic sensor and ultraviolet sensor implemented on a fire fighter robot using AT89S52", Journal of Robotics and Control (JRC,2020), Volume 2, pp. 55- 58.
- [14] S. Jakthi Priyanka, R. Sangeetha, "Android controlled firefighting robot", International Journal Of Innovative Science Engineering And Technology (IJISSET,2017), Volume 3.
- [15] Tawfiqur Rakib,M. A. Rashid Sarkar , "Design and fabrication of an autonomous firefighting robot with multi sensor fire detection using PID controller", ICIEV (2016), Volume 23, issue-1.

APPENDIX

A. SOURCE CODE

Module 1: Data Preparation using Web Scrapping

```
import os

import re

import cv2

import glob

import random

import shutil

import requests

from PIL import Image

from bs4 import BeautifulSoup

import matplotlib.pyplot as plt

import matplotlib.image as mpimg

%matplotlib inline

base_url =

'https://stock.adobe.com/search/images?hide_panel=true&k={}&search_type=u

sertyped'

def download_image(keyword, dl_path, base_url=base_url):

    url = base_url.format(keyword)

    r = requests.get(url)

    soup = BeautifulSoup(r.content, 'html.parser')

    # Some comment or placeholder text
```

```

links = soup.find_all('img', src=re.compile('.jpg'))

os.chdir(dl_path)

for link in links:

    src = link['src']

    with open(os.path.basename(src), 'wb') as f:

        f.write(requests.get(src).content)

from keywords import fire_keywords, night_keywords

print('Fire keywords: \n', fire_keywords,

      '\n\nNight keywords: \n', night_keywords)

dl_path = 'C:\\Users\\saiku\\Fire_Fighting_Robot\\Fire_set_dwn'

download_image('bonfire', dl_path, base_url=base_url)

for word in fire_keywords:

    download_image('fire+' + word, dl_path=dl_path)

dl_path = 'C:\\Users\\saiku\\Fire_Fighting_Robot\\Nofire_set_dwn'

for word in night_keywords:

    download_image('night+' + word, dl_path=dl_path)

base_dir = 'C:\\Users\\saiku\\Fire_Fighting_Robot'

src_fire = glob.glob(base_dir + '/Fire_set_*.')

src_nofire = glob.glob(base_dir + '/Nofire_set_*.')

dst_fire = os.path.join(base_dir, 'fire_images')

dst_nofire = os.path.join(base_dir, 'Nofire_images')

if not os.path.exists(dst_fire):

```

```

os.makedirs(dst_fire)

if not os.path.exists(dst_nofire):
    os.makedirs(dst_nofire)

def copy_file(src_path, dst_path, shuffle=False, no_modify=True, \
             start_index=None, end_index=None):
    src_files = os.listdir(src_path)

    if shuffle==True:
        for _ in range(10):
            random.shuffle(src_files)

    os.chdir(dst_path)

    for fname in src_files[start_index : end_index]:
        src = os.path.join(src_path, fname)

        if no_modify:
            dst = os.path.join(dst_path, fname)
            shutil.copyfile(src, dst)
        else:
            img = Image.open(src)

            if img.mode != 'RGB':
                if img.mode == 'P':
                    img = img.convert('RGBA').convert('RGB')
                else:
                    img = img.convert('RGB')

```

```

        if fname.endswith('.jpg'):

            dst = os.path.join(dst_path, fname)

            shutil.copyfile(src, dst)

        else:

            fn, fext = os.path.splitext(fname)

            img.save('{} .jpg'.format(fn))

def file_rename(path, name, n_shuffle=10):

    files = os.listdir(path)

    for _ in range(n_shuffle):

        random.shuffle(files)

    for index, file in enumerate(files):

        src = os.path.join(path, file)

        dst = os.path.join(path, \

            ".join([name, '.', str(index + 1), '.jpg']"))

        os.rename(src, dst)

file_rename(dst_fire, 'fire', n_shuffle=100)

file_rename(dst_nofire, 'nofire', n_shuffle=100)

train_dir = os.path.join(base_dir, 'train')

train_dir_fire = os.path.join(train_dir, 'fire')

if not os.path.exists(train_dir_fire):

    os.makedirs(train_dir_fire)

train_dir_nofire = os.path.join(train_dir, 'nofire')

```

```

if not os.path.exists(train_dir_nofire):
    os.makedirs(train_dir_nofire)

test_dir = os.path.join(base_dir, 'test')

test_dir_fire = os.path.join(test_dir, 'fire')

if not os.path.exists(test_dir_fire):
    os.makedirs(test_dir_fire)

test_dir_nofire = os.path.join(test_dir, 'nofire')

if not os.path.exists(test_dir_nofire):
    os.makedirs(test_dir_nofire)

n_23 = 3 * len(os.listdir(dst_fire)) // 4

copy_file(dst_fire, train_dir_fire, end_index=n_23)

copy_file(dst_nofire, train_dir_nofire, end_index=n_23)

copy_file(dst_fire, test_dir_fire, start_index=n_23)

copy_file(dst_nofire, test_dir_nofire, start_index=n_23)

def image_collage(path, n_rows, n_cols, title):
    fig = plt.figure()

    plt.title(title)

    plt.axis('Off')

    img_list = os.listdir(path)

    img_num = 1

    for img in img_list[:n_rows*n_cols]:
        img_path = os.path.join(path, img)

        img_read = mpimg.imread(img_path)

```

```

ax = fig.add_subplot(n_rows, n_cols, img_num)

img_num += 1

ax.imshow(img_read)

ax.axis('Off')

image_collage(train_dir_fire, 3, 4, 'Example Fire Images')

image_collage(train_dir_nofire, 3, 4, 'Example Non-Fire Images')

```

Module 2: Training the Data

```

from keras.applications import VGG16                # pretrained CNN

from keras.callbacks import ModelCheckpoint          # further train the saved
model

from keras import models, layers, optimizers        # building DNN is keras

from tensorflow.keras.models import load_model      # load saved model

from keras.preprocessing.image import ImageDataGenerator # preparing image
data for training

conv_base = VGG16(weights='imagenet',

                    include_top=False,

                    input_shape=(150, 150, 3))

model = models.Sequential()

model.add(conv_base)

model.add(layers.Flatten())

model.add(layers.Dense(256, activation='relu'))

```

```

model.add(layers.Dense(1, activation='sigmoid'))

conv_base.trainable = False

model.compile(loss='binary_crossentropy',
              optimizer=optimizers.RMSprop(lr=1e-4),
              metrics=['acc'])

model.summary()

# Set Data Directories

import os

base_dir = 'C:\\Users\\saiku\\Fire_Fighting_Robot'

train_dir = os.path.join(base_dir, 'train')

test_dir = os.path.join(base_dir, 'test')

train_datagen = ImageDataGenerator(rescale=1./255,
                                   rotation_range=40,
                                   width_shift_range=0.2,
                                   height_shift_range=0.2,
                                   shear_range=0.2,
                                   zoom_range=0.2,
                                   horizontal_flip=True)

test_datagen = ImageDataGenerator(rescale=1./255)

train_generator = train_datagen.flow_from_directory(train_dir,
                                                    target_size=(150, 150),
                                                    batch_size=32,

```

```

class_mode='binary')

test_generator = test_datagen.flow_from_directory(test_dir,

                                                target_size=(150, 150),

                                                batch_size=32,

                                                class_mode='binary')

history = model.fit(train_generator, epochs=30,

                    validation_data=test_generator)

model.save('VGG16_lr-4.h5')

# Dictionary to extract the numbers

hist_dict = history.history

# Training and validation accuracy

training_acc = hist_dict['acc']

validation_acc = hist_dict['val_acc']

# Training and validation loss

training_loss = hist_dict['loss']

validation_loss = hist_dict['val_loss']

# Number of epochs

epoches = range(1, 1 + len(training_acc))

import numpy as np                # linear algebra

from PIL import Image             # Python image library

import matplotlib.pyplot as plt   # making plots

%matplotlib inline

from IPython.display import display    # displaying ?

```

```

import warnings                                # ignoring unnecessary python warnings

warnings.filterwarnings('ignore')

def plot_func(entity):
    """
    This function produces plot to compare the performance
    between train set and validation set.
    entity can be loss or accuracy.
    """

    plt.figure(figsize=(8, 5))

    plt.plot(epochs, eval('training_' + entity), 'r')

    plt.plot(epochs, eval('validation_' + entity), 'b')

    plt.legend(['Training ' + entity, 'Validation ' + entity])

    plt.xlabel('Epochs')

    plt.ylabel(entity)

    plt.show()

plot_func('loss')

plot_func('acc')

conv_base.trainable = True

set_trainable = False

for layer in conv_base.layers:

    if layer.name == 'block5_conv1':

        set_trainable = True

    else:

```

```

set_trainable = False

model.compile(loss='binary_crossentropy',
              optimizer=optimizers.RMSprop(lr=1e-5),
              metrics=['acc'])

history = model.fit_generator(train_generator, epochs=50,
                             validation_data=test_generator)

model.save('VGG16_fine_tuned.h5')

hist_dict = history.history

training_accuracy = hist_dict['acc']

validation_accuracy = hist_dict['val_acc']

training_loss = hist_dict['loss']

validation_loss = hist_dict['val_loss']

epoches = range(1, 1 + len(training_acc))

from keras.callbacks import ModelCheckpoint

from tensorflow.keras.models import load_model

import matplotlib.pyplot as plt

def plot_func(entity):

    # Ensure lengths are consistent

    min_len = min(len(epoches), len(training_loss), len(validation_loss))

    epoches_trimmed = epoches[:min_len]

    training_entity_trimmed = eval('training_' + entity)[:min_len]

    validation_entity_trimmed = eval('validation_' + entity)[:min_len]

```

```

plt.figure(figsize=(8, 5))

plt.plot(epochs_trimmed, training_entity_trimmed, 'r')

plt.plot(epochs_trimmed, validation_entity_trimmed, 'b')

plt.legend(['Training ' + entity, 'Validation ' + entity])

plt.xlabel('Epochs')

plt.ylabel(entity.capitalize())

plt.title('Training and Validation ' + entity.capitalize())

plt.show()

# Call the function with 'loss' or 'acc'

plot_func('loss')

plot_func('accuracy')

# Loading the saved model

model = load_model('VGG16_fine_tuned.h5')

# taking first batch from the generator

img, label = test_generator[0]

# Predicting the images from the first batch

pred = np.round(model.predict(img)).flatten()

len(img)

# Numeric to semantic labels

label_dict = {1.0: 'No fire', 0.0: 'Fire'}

# Generating collage of plots

fig = plt.figure(figsize=(10, 9))

plt.title('Classification by the model')

```

```

plt.axis('off')

for i, img_i in enumerate(img[:20]):

    ax = fig.add_subplot(4, 5, i+1)

    plt.axis('off')

    plt.title(label_dict[pred[i]], y=-0.2)

    ax.imshow(img_i)

# Lists for missed fire images and missed non-fire images

msd_fire = []

msd_nofire = []

# Iterating through all the batches

for j in range(len(test_generator)):

    img, label = test_generator[j]

    pred = np.round(model.predict(img)).flatten()

    bool_list = label == pred

    # bool_list is False when there is misclassification

    for i, e in enumerate(bool_list):

        if e == False:

            # separating labels (fire and non-fire)

            if label[i] == 0:

                msd_fire.append(img[i])

            else:

                msd_nofire.append(img[i])

```

```

# total number of sample in train set in each class
n_class = 1001

# number of misclassified fire and non-fire images
nm_fire, nm_nofire = len(msd_fire), len(msd_nofire)

# confusion matrix (flattened)
conf_mat = [n_class-nm_fire, nm_fire, nm_nofire, n_class-nm_nofire]

# visualization of confusion matrix
fig = plt.figure(figsize=(4, 4))
for i, j in enumerate(conf_mat):
    ax = fig.add_subplot(2, 2, i+1)
    ax.imshow([[j]], vmin=0, vmax=1000, cmap='copper_r')
    ax.text(-0.2, 0.1, j, c='r', fontsize=30)
    ax.axis('off')

# bringing blocks tighter
fig.tight_layout()
fig.show()

fig = plt.figure(figsize=(10, 8))
plt.axis('off')

# Limit the number of images to 20 or the total number of images you have
num_images_to_display = min(len(msd_fire), 20)
for i, img_i in enumerate(msd_fire[:num_images_to_display]):
    ax = fig.add_subplot(4, 5, i+1)

```

```

    ax.imshow(img_i)

    ax.axis('off')

plt.show()

fig = plt.figure(figsize=(10, 4))

plt.axis('off')

# Limit the number of images to 10 or the total number of images you have
num_images_to_display = min(len(msd_nofire), 10)

for i, img_i in enumerate(msd_nofire[:num_images_to_display]):

    ax = fig.add_subplot(2, 5, i+1)

    ax.imshow(img_i)

    ax.axis('off')

plt.show()

checkpoint = ModelCheckpoint('VGG16_fine_tuned.h5',

                             monitor='loss', verbose=1,

                             save_best_only=True, mode='min')

callbacks_list = [checkpoint]

hst_cntd = model.fit(train_generator, epochs=5,

                     steps_per_epoch = 15,

                     validation_data=test_generator,

                     validation_steps = 30,

                     callbacks=callbacks_list)

```

Module 3: Real Time Processing

```
from __future__ import absolute_import

from __future__ import division

from __future__ import print_function

import argparse

import sys

import tensorflow as tf

from scipy import misc

import cv2

import numpy as np

import facenet

import detect_face

import os

import time

import pickle

from utils import *

parser = argparse.ArgumentParser()

parser.add_argument('--model-cfg', type=str, default='./yolo_cfg/yolov3-face.cfg',

                    help='path to config file')

parser.add_argument('--model-weights', type=str,

                    default='./yolo_weights/yolov3-wider_16000.weights',

                    help='path to weights of model')

args = parser.parse_args()
```

```

print('Creating networks and loading parameters')

net = cv2.dnn.readNetFromDarknet(args.model_cfg, args.model_weights)

net.setPreferableBackend(cv2.dnn.DNN_BACKEND_OPENCV)

net.setPreferableTarget(cv2.dnn.DNN_TARGET_CPU)

with tf.Graph().as_default():

    gpu_options = tf.GPUOptions(per_process_gpu_memory_fraction=0.6)

    sess = tf.Session(config=tf.ConfigProto(gpu_options=gpu_options,
log_device_placement=False))

    with sess.as_default():

        # pnet, rnet, onet = detect_face.create_mtcnn(sess, './models/')

        minsize = 20 # minimum size of face

        threshold = [0.6, 0.7, 0.7] # three steps's threshold

        factor = 0.709 # scale factor

        margin = 44

        frame_interval = 3

        batch_size = 1000

        image_size = 182

        input_image_size = 160

        print('Loading feature extraction model')

        modeldir = './models/facenet/20190308-101738'

        facenet.load_model(modeldir)


images_placeholder = tf.get_default_graph().get_tensor_by_name("input:0")

```

```

embeddings = tf.get_default_graph().get_tensor_by_name("embeddings:0")

phase_train_placeholder =
tf.get_default_graph().get_tensor_by_name("phase_train:0")

embedding_size = embeddings.get_shape()[1]

classifier_filename = './myclassifier/my_classifier.pkl'

classifier_filename_exp = os.path.expanduser(classifier_filename)

with open(classifier_filename_exp, 'rb') as infile:

    (model, class_names) = pickle.load(infile)

    print('load classifier file-> %s' % classifier_filename_exp)

video_capture = cv2.VideoCapture(0)

c = 0

print('Start Recognition!')

prevTime = 0

while True:

    ret, frame = video_capture.read()

    # frame = cv2.resize(frame, (0,0), fx=0.5, fy=0.5)  #resize frame
(optional)

    curTime = time.time()  # calc fps

    timeF = frame_interval

    if (c % timeF == 0):

        find_results = []

        if frame.ndim == 2:

```

```

        frame = facenet.to_rgb(frame)

frame = frame[:, :, 0:3]

#print(frame.shape[0])

#print(frame.shape[1])

# Use YOLO to get bounding boxes

blob = cv2.dnn.blobFromImage(frame, 1 / 255, (IMG_WIDTH,
IMG_HEIGHT), [0, 0, 0], 1, crop=False)

# Sets the input to the network

net.setInput(blob)

# Runs the forward pass to get output of the output layers

outs = net.forward(get_outputs_names(net))

# Remove the bounding boxes with low confidence

bounding_boxes = post_process(frame, outs, CONF_THRESHOLD,
NMS_THRESHOLD)

nrof_faces = len(bounding_boxes)

## Use MTCNN to get the bounding boxes

# bounding_boxes, _ = detect_face.detect_face(frame, minsize, pnet,
rnet, onet, threshold, factor)

# nrof_faces = bounding_boxes.shape[0]

#print('Detected_FaceNum: %d' % nrof_faces)

if nrof_faces > 0:

    # det = bounding_boxes[:, 0:4]

    img_size = np.asarray(frame.shape)[0:2]

```

```

# cropped = []

# scaled = []

# scaled_reshape = []

bb = np.zeros((nrof_faces,4), dtype=np.int32)

for i in range(nrof_faces):

    emb_array = np.zeros((1, embedding_size))

    bb[i][0] = bounding_boxes[i][0]

    bb[i][1] = bounding_boxes[i][1]

    bb[i][2] = bounding_boxes[i][2]

    bb[i][3] = bounding_boxes[i][3]

    if bb[i][0] <= 0 or bb[i][1] <= 0 or bb[i][2] >= len(frame[0]) or bb[i][3]
>= len(frame):

        print('face is inner of range!')

        continue

    # cropped.append(frame[bb[i][1]:bb[i][3], bb[i][0]:bb[i][2], :])

    # cropped[0] = facenet.flip(cropped[0], False)

    # scaled.append(misc.imresize(cropped[0], (image_size,
image_size), interp='bilinear'))

    # scaled[0] = cv2.resize(scaled[0],
(input_image_size,input_image_size),

    #             interpolation=cv2.INTER_CUBIC)

    # scaled[0] = facenet.prewhten(scaled[0])

    # scaled_reshape.append(scaled[0].reshape(-
1,input_image_size,input_image_size,3))

```

```

        # feed_dict = {images_placeholder: scaled_reshape[0],
phase_train_placeholder: False}

        cropped = (frame[bb[i][1]:bb[i][3], bb[i][0]:bb[i][2], :])

        print("{0} {1} {2} {3}".format(bb[i][0], bb[i][1], bb[i][2], bb[i][3]))

        cropped = facenet.flip(cropped, False)

        scaled = (misc.imresize(cropped, (image_size, image_size),
interp='bilinear'))

        scaled = cv2.resize(scaled, (input_image_size,input_image_size),

            interpolation=cv2.INTER_CUBIC)

        scaled = facenet.prewhten(scaled)

        scaled_reshape = (scaled.reshape(-
1,input_image_size,input_image_size,3))

        feed_dict = {images_placeholder: scaled_reshape,
phase_train_placeholder: False}

        emb_array[0, :] = sess.run(embeddings, feed_dict=feed_dict)

        predictions = model.predict_proba(emb_array)

        best_class_indices = np.argmax(predictions, axis=1)

        best_class_probabilities =
predictions[np.arange(len(best_class_indices)), best_class_indices]

        cv2.rectangle(frame, (bb[i][0], bb[i][1]), (bb[i][2], bb[i][3]), (0, 255,
0), 2)

        text_x = bb[i][0]

        text_y = bb[i][3] + 20

        # for H_i in HumanNames:

```

```

        # if HumanNames[best_class_indices[0]] == H_i:

        result_names = class_names[best_class_indices[0]]

        #print(result_names)

        cv2.putText(frame, result_names, (text_x, text_y),
cv2.FONT_HERSHEY_COMPLEX_SMALL,

                    1, (0, 0, 255), thickness=1, lineType=2)

    else:

        print('Unable to align')

    sec = curTime - prevTime

    prevTime = curTime

    fps = 1 / (sec)

    str = 'FPS: %2.3f' % fps

    text_fps_x = len(frame[0]) - 150

    text_fps_y = 20

    cv2.putText(frame, str, (text_fps_x, text_fps_y),

                cv2.FONT_HERSHEY_COMPLEX_SMALL, 1, (0, 0, 0),
thickness=1, lineType=2)

    # c+=1

    cv2.imshow('Video', frame)

    if cv2.waitKey(1) & 0xFF == ord('q'):

        break

video_capture.release()

cv2.destroyAllWindows()

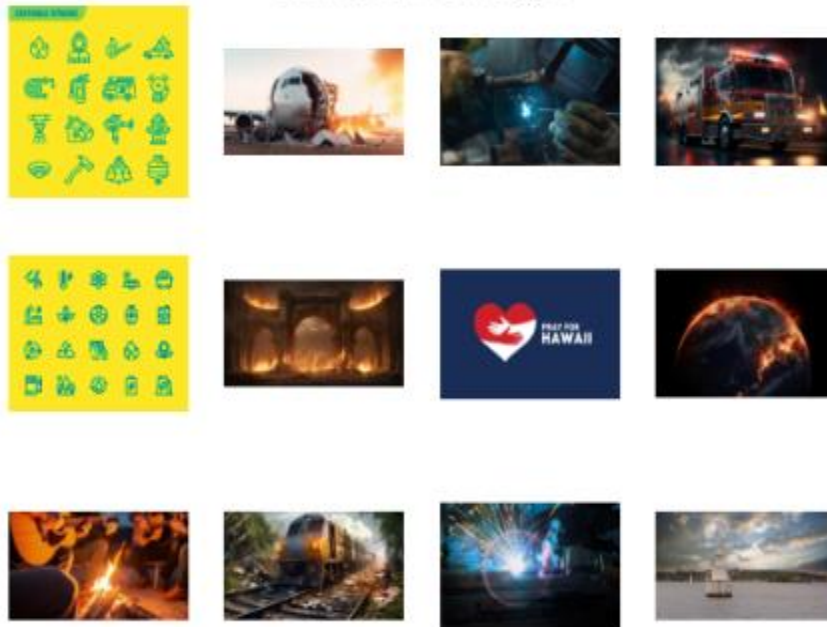
```

B. SCREENSHOTS

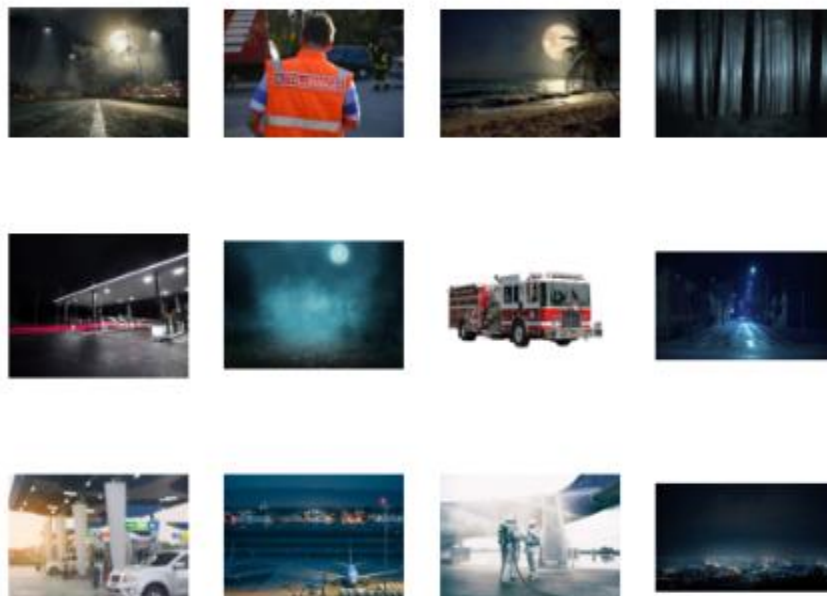
Data Collection using Web Scrapping

```
In [53]: image_collage(train_dir_fire, 3, 4, 'Example Fire Images')  
image_collage(train_dir_nofire, 3, 4, 'Example Non-Fire Images')
```

Example Fire Images

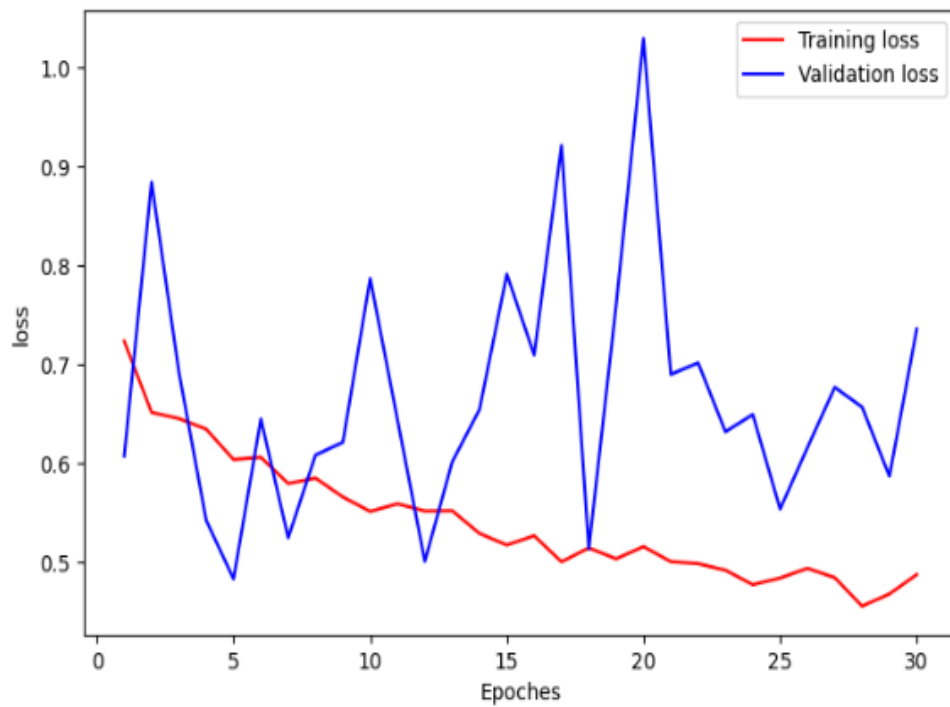


Example Non-Fire Images

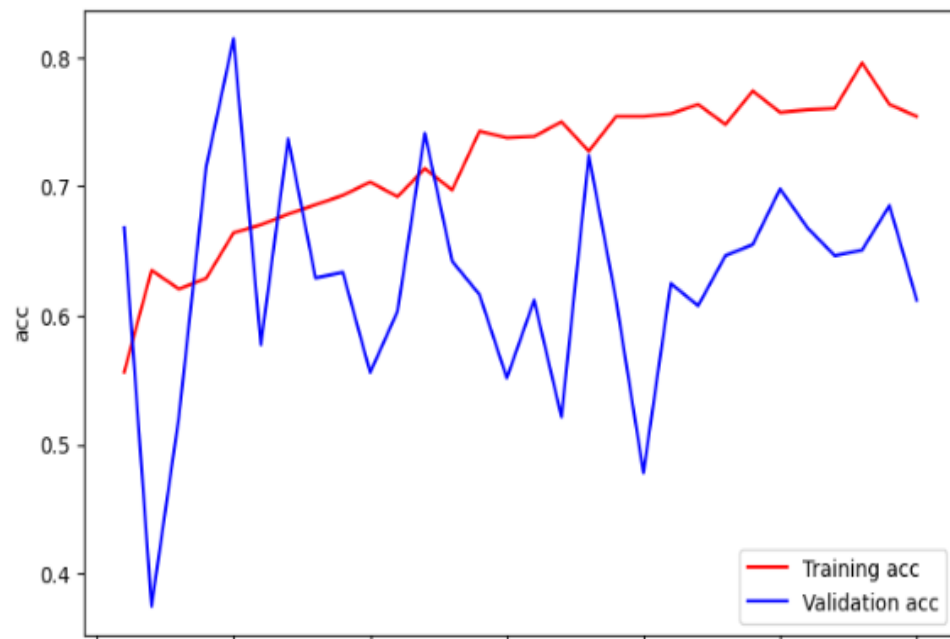


Accuracy and Loss Functions

```
In [13]: plot_func('loss')
```



```
In [14]: plot_func('acc')
```

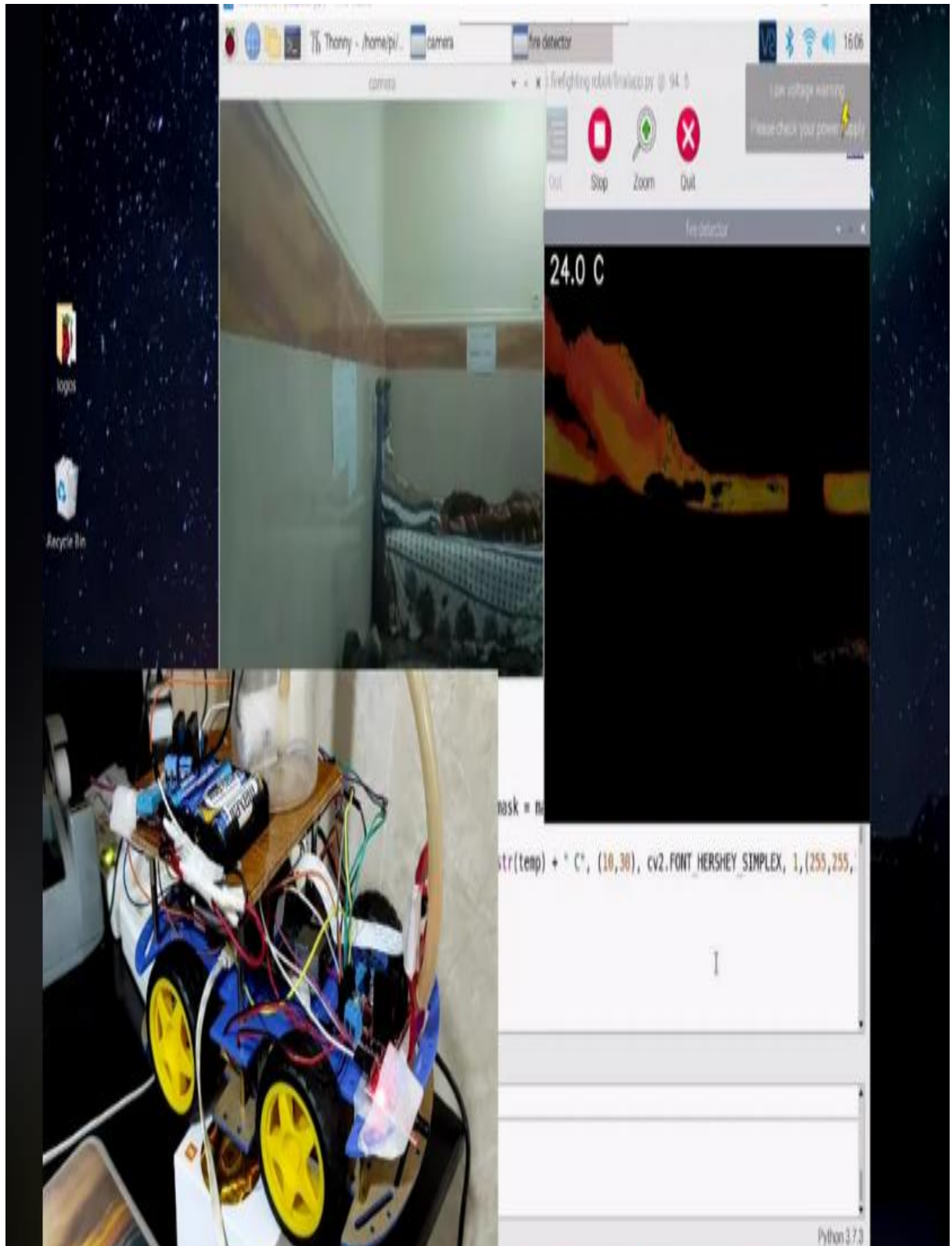


Epoch Values for Validating the trained Dataset

```
In [16]: history = model.fit_generator(train_generator, epochs=50,  
                                     validation_data=test_generator)
```

```
Epoch 1/50  
31/31 [=====] - 413s 13s/step - loss: 0.5035 - acc: 0.7482 - val_loss: 0.5895 - val_acc: 0.7069  
Epoch 2/50  
31/31 [=====] - 398s 13s/step - loss: 0.4235 - acc: 0.8021 - val_loss: 0.6802 - val_acc: 0.6595  
Epoch 3/50  
31/31 [=====] - 399s 13s/step - loss: 0.4053 - acc: 0.8135 - val_loss: 0.7680 - val_acc: 0.6509  
Epoch 4/50  
31/31 [=====] - 394s 13s/step - loss: 0.4081 - acc: 0.8073 - val_loss: 0.6978 - val_acc: 0.6509  
Epoch 5/50  
31/31 [=====] - 391s 13s/step - loss: 0.3914 - acc: 0.8145 - val_loss: 0.5408 - val_acc: 0.7155  
Epoch 6/50  
31/31 [=====] - 434s 14s/step - loss: 0.3832 - acc: 0.8311 - val_loss: 0.6612 - val_acc: 0.6767  
Epoch 7/50  
31/31 [=====] - 509s 16s/step - loss: 0.3842 - acc: 0.8124 - val_loss: 0.6331 - val_acc: 0.6983  
Epoch 8/50  
31/31 [=====] - 498s 16s/step - loss: 0.3356 - acc: 0.8487 - val_loss: 0.7730 - val_acc: 0.6810  
Epoch 9/50  
31/31 [=====] - 1377s 45s/step - loss: 0.3404 - acc: 0.8456 - val_loss: 0.6871 - val_acc: 0.7069  
Epoch 10/50  
31/31 [=====] - 513s 17s/step - loss: 0.3290 - acc: 0.8487 - val_loss: 0.7036 - val_acc: 0.6983  
Epoch 11/50  
31/31 [=====] - 598s 19s/step - loss: 0.3314 - acc: 0.8518 - val_loss: 0.6725 - val_acc: 0.7198  
Epoch 12/50  
31/31 [=====] - 602s 20s/step - loss: 0.3349 - acc: 0.8435 - val_loss: 0.7597 - val_acc: 0.7026  
Epoch 13/50  
31/31 [=====] - 507s 16s/step - loss: 0.3088 - acc: 0.8601 - val_loss: 0.6512 - val_acc: 0.7155  
Epoch 14/50  
31/31 [=====] - 587s 19s/step - loss: 0.3396 - acc: 0.8404 - val_loss: 0.8468 - val_acc: 0.6767  
Epoch 15/50  
31/31 [=====] - 503s 16s/step - loss: 0.3202 - acc: 0.8549 - val_loss: 0.7315 - val_acc: 0.7112  
Epoch 16/50  
31/31 [=====] - 509s 16s/step - loss: 0.2984 - acc: 0.8611 - val_loss: 0.7032 - val_acc: 0.7328  
Epoch 17/50  
31/31 [=====] - 519s 17s/step - loss: 0.2891 - acc: 0.8518 - val_loss: 0.7697 - val_acc: 0.7069  
Epoch 18/50  
31/31 [=====] - 502s 16s/step - loss: 0.2683 - acc: 0.8777 - val_loss: 0.6254 - val_acc: 0.7672  
Epoch 19/50  
31/31 [=====] - 1091s 36s/step - loss: 0.2879 - acc: 0.8580 - val_loss: 0.5789 - val_acc: 0.7500  
Epoch 20/50  
31/31 [=====] - 515s 17s/step - loss: 0.3033 - acc: 0.8508 - val_loss: 0.8156 - val_acc: 0.6897  
Epoch 21/50  
31/31 [=====] - 2570s 85s/step - loss: 0.2848 - acc: 0.8798 - val_loss: 0.7949 - val_acc: 0.6983  
Epoch 22/50  
31/31 [=====] - 405s 13s/step - loss: 0.2905 - acc: 0.8777 - val_loss: 0.6578 - val_acc: 0.7457  
Epoch 23/50  
31/31 [=====] - 387s 12s/step - loss: 0.2567 - acc: 0.8829 - val_loss: 0.6620 - val_acc: 0.7414  
Epoch 24/50  
31/31 [=====] - 407s 13s/step - loss: 0.2631 - acc: 0.8860 - val_loss: 0.6905 - val_acc: 0.7414
```

Real time Fire Detection and Controlling



A Machine Vision based Fire Fighting Robot with Real-Time Fire Localization

Mr. K. D. N. SAI KUMAR
Department of Computer Science
and Technology
Sathyabama Institute of Science
and Technology
Chennai, India
Saikumarkatepalli89@gmail.com

Mr. K. KRISHNA MADHAVA REDDY
Department of Computer Science
and Technology
Sathyabama Institute of Science
and Technology
Chennai, India
krishnamadhav735000@gmail.com

Dr. NANCY NOELLA R S
Department of Computer Science
and Technology
Sathyabama Institute of Science
and Technology
Chennai, India
nancynoella.cse@sathyabama.ac.in

ABSTRACT

The incorporation of deep learning algorithms, particularly YOLOv8, into an intelligent robotic system aimed at detecting and pinpointing fires in camera imagery. Our method has achieved a noteworthy 79% accuracy in classifying fire detection, representing a significant achievement in real-time object recognition. While recognizing this achievement, the research acknowledges significant opportunities for further enhancement in the detection of fire types. The implementation of machine vision in the advanced robot establishes the ground-work for adaptability in response to different fire situations. This initial milestone signifies a promising advancement in the direction of more efficient fire response, and also highlights the ongoing progress in the convergence of deep learning and robotics. The results suggest practical uses for intelligent robots in real-life scenarios, particularly in emergency response and public safety situations.

Keywords: You Only Look Once (YOLO)

I. INTRODUCTION

The intersection of advanced technologies has spurred revolutionary advancements across a range of fields, with important implications for essential applications. In this particular setting, the current study focuses on the creation of a robot designed to control fires, using the effective YOLOv8 algorithm in combination with the computational capabilities of a Raspberry Pi platform. The emergence of deep learning algorithms, exemplified by the You Only Look Once (YOLO) approach, has significantly transformed the field of computer vision, especially in the area of real-time object detection.

This study aims to leverage the capabilities of YOLOv8 for immediate fire detection and accurate pinpointing, introducing a new approach for implementation on the re-source-efficient and widely available Raspberry Pi architecture. The convergence of YOLOv8 and Raspberry Pi not only ensures timely and precise object recognition but also envisages a practical and cost-efficient solution for enhancing fire control systems. This introduction provides the justification for examining the potential synergies between YOLOv8 and Raspberry Pi, with the goal of enhancing the capabilities of fire-controlling robots and making a significant contribution to the development of intelligent systems in emergency response situations.

II. RELATED WORKS

Ligang Chen [1] introduced a model centered around the stm32f103zet6 MCU, chosen for its efficacy in low power consumption and robust performance. This particular model integrates a portable fire extinguisher. The robot is outfitted with a camera in its head, enabling image capture and data collection. To facilitate wireless communication, the robot employs the NRF24L01 wireless transmission module.

A. Hassanein et al. [2] suggested a model that involves the utilization of both a PIC microcontroller and an Arduino Mega microcontroller, along with the incorporation of a supplementary Bluetooth module. The robot's guidance is facilitated by a digital compass GY-26, aiding in determining the degree of rotation from its initial position. The specific Bluetooth module employed, the Kootek BT2s, interfaces with MATLAB. Notably, a significant drawback identified in this model was the lack of precision in the robot's movement.

M.A. Hossain et al. [3] introduced a model comprising a camera for data collection alongside various sensors. This combined setup is responsible for gathering data and subsequently transmitting it to the Node MCU, which then relays the information to a server. Following an analysis of the situation based on a programmed protocol, the resulting information is forwarded to the controller.

M. Kanwar and L. Agilandeewari [4] introduced a model wherein the robot transmits a fire alert to the cloud, pinpointing the fire's location. An accompanying application utilizes this data to provide individuals with an exit route out of the building. Furthermore, sensors integrated into the system provide in-

formation regarding the level of carbon dioxide. Authorized personnel can then manually choose between water or CO₂ for extinguishing the fire based on this data.

L. Mingsong and L. Tugan [5] developed a model, includes a 360-degree rotatable camera to enable continuous video streaming. Additionally, IR sensors have been integrated to detect obstacles while the robot is in motion. This model provides both manual and automatic control features.

Ambadkar and colleagues [6] proposed the model which involves the Arduino functioning as the primary control unit for the robot. It connects with a fire detection system equipped with temperature and gas sensors. Furthermore, the robot is outfitted with a wireless camera that transmits the robot's front-facing view to a receiving unit consisting of an XBEE 3, an Arduino computer, and a camera receiver. Utilizing a graphical user interface (GUI), an operator has the ability to visually assess the front view and issue commands that dictate the actions of the robot. Additionally, the robot is equipped with a motor driving unit that is controlled by an Arduino, a sprinkler pump for the purpose of extinguishing fires, and a relay driver circuit.

Prasojo et al. [7] devised a fire controlling robot, leveraging the AT89S52 as its controller. A relay was employed for extinguishing fires, and a fire sensor was utilized to detect their presence. The robot utilized a DC motor for movement and featured an ultrasound-based navigation system to detect surrounding obstacles. Upon obstacle detection by the ultrasound system, the robot could identify fires within a range of up to 5 meters and effectively extinguish them.

AlHaza et al. [8] proposed that the firefighting robot was designed to enter highly

hazardous fire environments, combat fires internally, and perform rescue operations. Capable of navigating stairs and enduring temperatures of up to 700 degrees Celsius for approximately 60 minutes, this robot employed multiple thermal isolation techniques. It provided gas masks and oxy-gen bottles for trapped individuals, incorporated two built-in fire extinguishing cylinders, featured flame detection sensors, and was equipped with three mounted IR camera units and analysis of data pertaining to cognitive functions and motor activity through the use of sensor-based technologies and machine learning techniques.

Dhumatkar et al. [9] proposed a model incorporating a thermostat to sense system temperatures, regulate heating or cooling devices, or control heat transfer fluid flow to maintain specific temperatures. A DC motor adjusts voltage through on-off cycles at low voltages. Additionally, a pump facilitated fluid movement through mechanical action, and wireless remote navigation was achieved using IC7442. Furthermore, a wireless camera enabled live demonstrations in remote locations where human presence was not feasible.

Anantha Raj P et al. [10] presented a model comprising multiple nodes, each equipped with an array of sensors and an Arduino microcontroller. This central coordinator, operated by a Raspberry Pi processor, assumes control of the entire Indoor Fire-fighting and Rescue Team (IORT) system. Upon detection of a fire by any node, the respective node notifies the central coordinator. Subsequently, the central coordinator relays this information to fire safety officers and triggers the mobile firefighting robot to initiate fire-fighting procedures.

Fan et al. [11] introduced a model that integrates the Gmapping SLAM algorithm with a

fire source identification image processing algorithm. This combination enables the fire-fighting robot to autonomously navigate and detect fire sources. The SLAM construction involves the RBPF algorithm, with a specific focus on utilizing an optimal number of particles for effective mapping. Meanwhile, the using a series of image processing methods, fire sources can be detect-ed and located.

Tawfiqur Rakib. [12] suggested a fire-fighting robot design with a base made of 'Kerosene wood'. It has an LM35 sensor to measure temperature changes, flame sensors to find fires, and a 1-liter water tank made of strong, waterproof cardboard. The robot's movement relies on two wheels.

S. Jakthi Priyanka. [13] proposed an fire-fighting robot that can be controlled by an Android device using the Arduino UNO R3. The robot has a gas sensor to find fires, a gear motor and a motor drive to move around, and a Bluetooth module to connect with an Android device for smartphone control. It also has a water pump and a sprinkler system. The Arduino UNO is programmed using the free Arduino IDE software to write and execute functions in the microcontroller.

Nagesh MS. [14] suggested a fire-fighting robot that can be controlled by DTMF (Dual Tone Multi Frequency Tones) technology for moving around. The robot has a flame sensor that can find fires within the wavelength range of 760 to 1100 nm, with a sensitivity range from 10cm to 1.5 feet.

Khaled Sailan. [15] suggested a robot that can avoid obstacles on land and water using a fuzzy controller. This robot can steer and dodge fixed obstacles in real time, making sure it follows its planned route while avoiding any obstacles on its way.

Table 1 Summary of literature survey

Author(s)	Title	Proposal	Advantages	Disadvantages	Proposed Work
Ligang Chen	Firefighting Robot with stm32f103zet6 MCU	A robot that uses stm32f103zet6 MCU as the main controller, which is low-power and strong. It has a portable fire extinguisher. The robot's head has a camera for taking pictures and gathering data. It uses NRF24L01 wireless module for communication.	Efficient and powerful MCU, low power consumption, portable fire extinguisher.	Limited information about the model's accuracy and capabilities.	Enhancing the navigation algorithms for precise and efficient firefighting operations.
Hassanein	Firefighting Robot with PIC and Arduino	A model using a PIC microcontroller and Arduino Mega with an additional Bluetooth module. A digital compass guides the robot. The Bluetooth module interfaces with MATLAB. The model faces issues with accurate movement.	Integration of PIC, Arduino, and Bluetooth for control.	Accuracy issues with robot movement.	Implementing advanced control algorithms to improve the accuracy of the robot's movement.
M.A. Hossain	Firefighting Robot with Camera and Sensors	A model with a camera and sensors sending data to NodeMCU, then to the server. Situation analyzed programmatically, and information sent to the controller.	Data collection through sensors and camera, server analysis.	Limited details about the specific sensors used.	Upgrading sensor technologies for enhanced data accuracy and exploring advanced analytics for improved situation analysis.
M. Kanwar and L. Agilandeeswari	Cloud-Connected Fire Alert Robot	The robot sends a fire alert to the cloud, providing fire location for an exit route. Sensors gauge carbon dioxide levels for manual fire extinguishing decisions.	Cloud connectivity, real-time fire location data.	Limited information on the types of sensors used.	Investigating cutting-edge sensor technologies for more accurate data and exploring machine learning algorithms for intelligent decision-making.
L. Mingsong and L. Tugan	Multi-Functional Firefighting Robot	Equipped with a 360-degree rotatable camera and IR sensors for obstacle detection. Manual and automatic control options.	Versatility with manual and automatic control.	Specifics on obstacle detection capabilities are unclear.	Advancing obstacle detection features through the incorporation of AI-based algorithms for improved safety.
Ambadkar	Arduino-Controlled Firefighting Robot	Arduino-controlled robot with fire sensing unit, wireless camera, motor driving unit, sprinkler pump, and relay circuit.	Arduino-based control system with various functionalities.	Limited details on the robot's autonomous decision-making.	Enhancing the autonomy of the robot by integrating advanced decision-making algorithms and improving communication capabilities.

Author(s)	Title	Proposal	Advantages	Disadvantages	Proposed Work
Prasojo	AT89S52-Based Fire Extinguisher Robot	Designed using AT89S52 as a controller, with relay for fire extinguishing. DC motor for movement, ultrasound-based navigation, obstacle detection, and fire detection up to 5 meters.	Utilization of AT89S52 for control, ultrasound navigation.	Limited information on the ultrasound-based navigation system.	Enhancing the ultrasound-based navigation system for better obstacle detection and exploring advanced firefighting strategies.
AlHaza	Advanced Firefighting Robot	A robot capable of entering dangerous fires, climbing stairs, withstanding high temperatures, and providing safety equipment. Equipped with fire extinguishing cylinders and IR cameras.	Capable of high-temperature operation and complex environments.	Complexity and maintenance challenges in extreme conditions.	Researching material sciences for improved robot durability and exploring self-maintenance mechanisms for extreme conditions.
Dhumatkar	Thermostat-Controlled Firefighting Robot	A model using a thermostat for temperature sensing, DC motor for adjustments, and a pump for fluid flow. IC7442 for wireless navigation and a wireless camera for live demonstration.	Precise temperature control with thermostat.	Limited information on the specific thermostat and pump used.	Investigating advanced thermostat technologies and pump systems for enhanced temperature control and fluid flow management.
Anantha Raj P	Wireless Sensor Network for Fire Safety	Nodes with sensors and Arduino microcontroller placed in locations prone to fire accidents. Connected to a central coordinator (Raspberry Pi) through wireless communication.	Extensive sensor network for fire detection and central coordination.	Details about specific sensor nodes and their capabilities are unclear.	Developing advanced sensor nodes with improved fire detection capabilities, enhancing central coordination system for real-time decision-making.
Fan et al	SLAM-Based Autonomous Firefighting Robot	A method that uses Gmapping SLAM algorithm and fire source identification image processing algorithm to make a robot move by itself and find fires. SLAM construction uses RBPF algorithm and effective particle number.	Autonomous navigation and fire source identification.	Limited details on SLAM construction and image processing techniques.	Advancing SLAM algorithms and image processing techniques for more accurate navigation and precise fire source identification.
Tawfiqur Rakib, M. A. Rashid Sarkar	Kerosene Wood-Based Firefighting Robot	A fire-fighting robot that has a base made of 'Kerosene wood,' a temperature sensor, fire detectors, and a water tank.	Eco-friendly base platform and sensors for fire detection.	Limited information on the robot's mobility and firefighting efficiency.	Researching innovative materials for enhanced mobility and firefighting efficiency, and optimizing water delivery mechanisms.

Author(s)	Title	Proposal	Advantages	Disadvantages	Proposed Work
S. Jakthi Priyanka. Sangeetha	Android-Controlled Firefighting Robot	Arduino UNO R3 controlled robot with gas sensor, Bluetooth module, water pump, and sprinkler. Controlled via an Android device using Arduino IDE.	Android-controlled robot with various functionalities.	Limited information on the gas sensor's sensitivity and range.	Improving gas sensor capabilities for better fire detection and exploring advanced control interfaces for enhanced user experience.
Nagesh MS, Deepika T V, Stafford Michahial, Dr M Sivakumar	fire-fighting robot that uses DTMF tones for control.	DTMF technology for robot navigation, flame sensor for fire detection, and central coordination for firefighting actions.	DTMF technology for efficient navigation.	Specifics on central coordination and firefighting actions are unclear.	Enhancing central coordination mechanisms and exploring advanced technologies for firefighting actions to improve the robot's overall effectiveness.
Khaled Sailan, Prof. Dr.-Ing. Klaus-Dieter Kuhnert, Simon Hardt	Fuzzy-Controlled Obstacle Avoidance Robot	An obstacle avoidance robot using a fuzzy controller to guide the robot along its path and avoid obstacles in real time.	Fuzzy control for real-time obstacle avoidance.	Limited details on the effectiveness of the fuzzy controller.	Researching and fine-tuning the fuzzy controller for optimal obstacle avoidance and exploring additional sensor technologies for improved path planning.

III. PROPOSED SYSTEM

In the Propose system, the YOLOv8 algorithm executes fire detection through a four-step process, seamlessly integrating Backbone, Neck, and Prediction stages.

Initially, In the "Backbone" phase, YOLOv8 establishes its core architecture using a potent neural network like Darknet, extracting hierarchical features from input images. The subsequent "Neck" refines these features, enhancing the model's ability to discern intricate details, crucial for recognizing specific characteristics in fire detection. The final "Prediction" stage utilizes learned features for accurate predictions. YOLOv8 divides the image into cells, predicting bounding boxes and class probabilities for real-time fire detection.

A. Hardware Configuration

Optimize YOLOv8 to function effectively on a Raspberry Pi, taking into account

computational limitations and prioritizing real-time processing. Guarantee a smooth integration of YOLOv8 with the hardware architecture of the Raspberry Pi to enable the efficient implementation of fire detection and suppression processes. Outfit the firefighting robot with strategically positioned high-resolution cameras and temperature sensors to ensure comprehensive coverage.

Create a strong, real-time communication system between the sensors and the Raspberry Pi, allowing for quick data collection

and analysis. Incorporate a dependable fire suppression system into the robot's design, such as a water spraying system or deployable fire extinguisher. Create synchronization algorithms to guarantee a prompt and coordinated reaction to identified fires, smoothly incorporating the abilities for fire detection and suppression.

B. Working

Utilize an advanced navigation system to enable the robot to independently navigate to identified fire locations. Integrate algorithms for obstacle avoidance to ensure safe navigation in dynamic and complex environments, thus improving the robot's ability to adapt during emergency situations. Create a user-friendly interface for real-time monitoring and control, which can be accessed through both web and mobile applications. Implement strong communication protocols to facilitate remote control and coordination with emergency response systems, thereby improving the robot's effectiveness and ability to respond promptly.

C. Fire Detection

Utilize YOLOv8 for the purpose of fire detection by employing pre-existing models and adapting them to a wide-ranging and annotated dataset encompassing a variety of fire situations. Modify the algorithm to effectively function in real-time on the Raspberry Pi, with a focus on maximizing both accuracy and computational efficiency. Perform thorough testing in controlled settings to assess the algorithm's precision in detecting various types of fires and their respective conditions. Confirm the strength and dependability of the fire detection system in different lighting and environmental conditions, ensuring its efficiency in real-world use.

D. Fire Extinguishing Process

Efficiently integrate the fire detection system with the fire suppression mechanism to ensure a timely and efficient response. Utilize algorithms to activate the fire suppression process upon detection, in order to enhance the response time and minimize potential harm.

Continuously improve the entire system by utilizing testing results and real-world implementations, adjusting parameters to achieve

improved performance. Establish continuous monitoring mechanisms in order to ensure that the fire-fighting robot is able to adapt to changing conditions, ensuring that timely updates are provided and emerging challenges in the field are addressed.

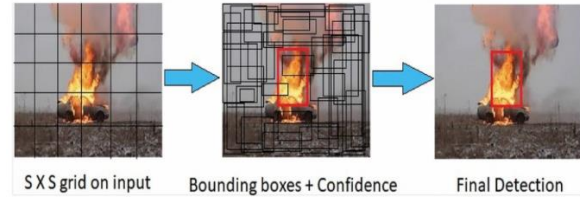


Fig. 1 Fire Detection

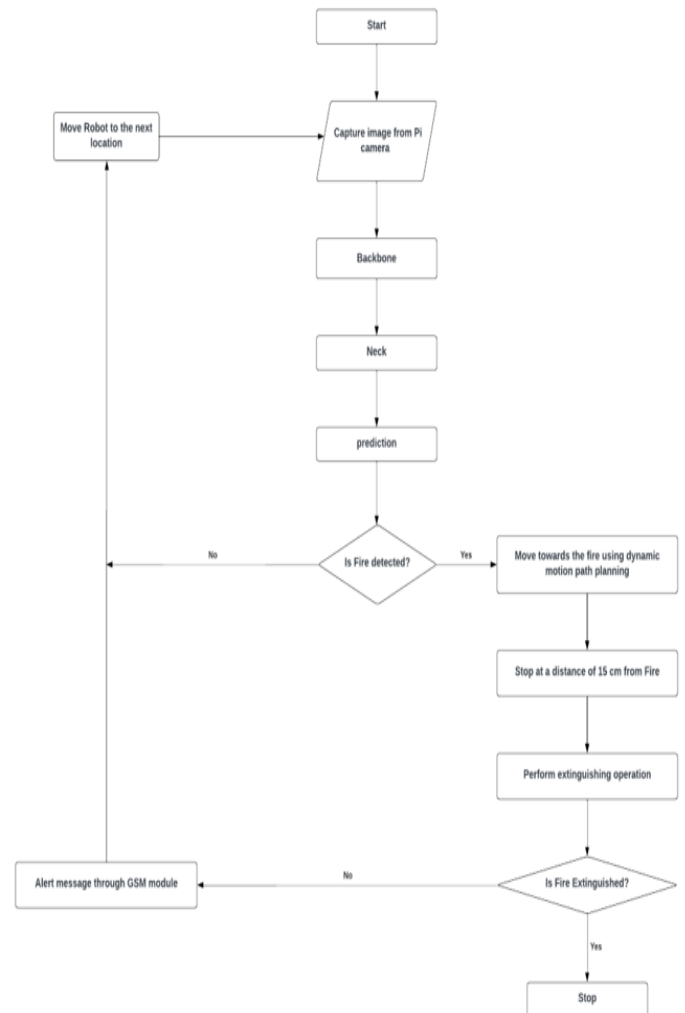


Fig. 2 Working Model

IV. CONCLUSION

The proposed system showcases an advanced implementation of the YOLOv8 algorithm in a fire-fighting robotic system. The efficient design incorporates DC motors, a dependable battery, Raspberry Pi 4, Pi Camera, and a specialized motor pump, showcasing effective navigation, immediate fire detection, and proactive involvement in fire suppression. The successful implementation of this integration demonstrates the potential for YOLOv8 to revolutionize emergency response systems. It presents significant developments in the integration of deep learning and robotics, which have the potential to improve the accuracy and efficiency of fire-fighting and other essential functions.

REFERENCES

- [1] A. Hassanein, M. Elhawary, N. Jaber and M. El-Abd, "An autonomous firefighting robot", 2015 International Conference on Advanced Robotics (ICAR, 2015), pp. 530-535, doi: 10.1109/ICAR.2015.7251507.
- [2] AlHaza, T., A. Alsadoon, Z. Alhusinan, M. Jarwali, and K. Alsaif. "New concept for indoor fire fighting robot", *Procedia-Social and Behavioral Sciences* 195 (2015), pp. 2343-2352.
- [3] Ambadkar, Komal N., Vaishnavee A. Gorte, Shravasti M. Rekhate, Renuka D. Nichit, Pratik A. Gaupal and P. K. Khedkar. "Fire Fighting Robot Using Arduino", *Int. Res. J. Eng. Technol* 6 (2019), pp. 3577-3578.
- [4] Anantha Raj and M. Srivani, "Internet of Robotic Things Based Autonomous Fire Fighting Mobile Robot," 2018 IEEE International Conference on Computational Intelligence and Computing Research (ICIC, 2018), pp. 1-4, doi: 10.1109/ICIC.2018.8782369.
- [5] Dhumatkar, Abhilash, Sumit Bhiogade, Shashank Rajpal, Datta Renge, and P. Kale. "Automatic fire fighting robot". *International Journal of Recent Research in Mathematics Computer Science and Information Technology* 2, no. 1 (2015), pp. 42-46.
- [6] Fan et al., "A Fire Protection Robot System Based on SLAM Localization and Fire Source Identification", 2019 IEEE 9th International Conference on Electronics Information and Emergency Communication (ICEIEC, 2019), pp. 555-560, doi: 10.1109/ICEIEC.2019.8784563.
- [7] Khaled Sailan, Klaus D. Kuhnert, and Hradik Karelia, "Modeling, Design and Implement of Steering Fuzzy PID Control System for DORIS Robot", *International Journal of Computer and Communication Engineering*, Vol. 3, No. 1 (2014).
- [8] L. Chen, "Design and Manufacture of Indoor Intelligent Fire Fighting Robot", 2020 International Workshop on Electronic Communication and Artificial Intelligence (IWECAI, 2020), pp. 201-204, doi: 10.1109/IWECAI50956.2020.00048.
- [9] L. Mingsong and L. Tugan, "Design and Experiment of Control System of Intelligent Fire Fighting Robot", 2020 IEEE 4th Information Technology, Networking, Electronic and Automation Control Conference (IT-NEC, 2020), pp. 2570-2573, doi: 10.1109/IT-NEC48623.2020.9084919
- [10] M. A. Hossain, H. S. Roy, M. F. K. Khondakar, M. H. Sarowar and M. A. Hossainline, "Design and Implementation of an IoT Based Firefighting and Affected Area Monitoring Robot", 2021 2nd International Conference on Robotics, Electrical and Signal Processing Techniques (ICREST, 2021), pp. 552-556.