Fire Detection using Sensor with Machine Learning

Presented by:

Under the Guidance:

Introduction

This Python-based fire detection system utilizes OpenCV and a Haar Cascade model to identify fire in real-time through a webcam feed. It incorporates threading for simultaneous alarm and email alerts without interrupting detection. The system also visualizes detection statistics using graphs such as detection count, frequency, and moving average. This solution is designed to be efficient ,automated , and informative for safety monitoring applications.

Existing System

- **Traditional Fire Detection Methods**:
  - ✓ Most systems rely on **smoke** or **heat sensors**.
  - ✓ Sensors trigger alarms only when certain thresholds are met.
- **Limitations**:
  - ✓ **Reactive**: Detects fire only after it reaches a specific stage (smoke or heat).
  - ✓ **Slow response**: May not respond quickly enough in open or large areas.
  - ✓ **False negatives**: Can miss early-stage fires, leading to delayed alerts.
- **Lack of Real-Time Monitoring**:
  - ✓ Often limited to certain areas of a building or environment.
  - ✓ Fire detection is not always available **24/7** for real-time monitoring.

Problem Statement

- Traditional fire detection systems respond slowly, increasing damage and danger.
- Early-stage fires often go undetected due to lack of visible smoke or heat.
- Manual monitoring causes delays—automated real-time alerts are essential.

Objectives

- Develop A Fire Detection System Using opencv and ML.
- This presentation explores the advancement of fire detection using sensor fusion and machine learning, highlighting its importance, architecture, challenges, and applications.

- **Integration of Real-Time Video Analysis**:
    - ✓ Leverage **OpenCV** for real-time video processing to detect fire visually through color and motion analysis, enabling instant detection and response.
- **Automated Alerts for Enhanced Safety**:
    - ✓ Incorporate **threading** to trigger alarms and send **email notifications** immediately when a fire is detected, ensuring timely intervention and increasing overall safety.

Key Features

- **Real-time Fire Detection:**
  - ✓ Uses live video feed and OpenCV to detect fire instantly through visual analysis.
- **Color-Based Fire Recognition:**
  - ✓ Detects fire-like regions using HSV color space and fire color thresholds for higher accuracy.
- **Confidence Scoring System:**
  - ✓ Calculates a confidence percentage to filter out false positives and ensure reliable detection.
- **Email Alert System:**
  - ✓ Sends an automated email notification to a registered address during a fire event for remote awareness.
- **Automated Alarm Sound:**
  - ✓ Plays an alarm automatically when fire is detected to alert nearby individuals immediately.
- **Customizable Alert Mechanism:**
  - ✓ Alert settings (e.g., sound file, recipient email, confidence threshold) can be easily modified for different environments or users.

System Requirements & Tools

- **Hardware Requirements**
  - ✓ Standard PC/Laptop
  - ✓ Built-in or External Webcam
  - ✓ **Internet Connection** *(only for sending email alerts)*
  - ✓ **Speakers** *(for alarm playback)*

- **Software Requirements**
  - ✓ **OS:** Windows / Linux / macOS
  - ✓ **Python : 3.x**
  - ✓ **Libraries :** Python, OpenCV, nympy, matplotlib, threading, smtp, Camera (Webcam/USB)

- **Tools & Technologies**
  - ✓ **Jupyter Notebook / VS Code / PyCharm / Spyder**– For coding and debugging
  - ✓ **Haar Cascade XML File** – Pre-trained model for fire detection
  - ✓ **Webcam** – For capturing live video stream
  - ✓ **Email Account** – To send real-time fire alerts

Machine Learning Algorithm

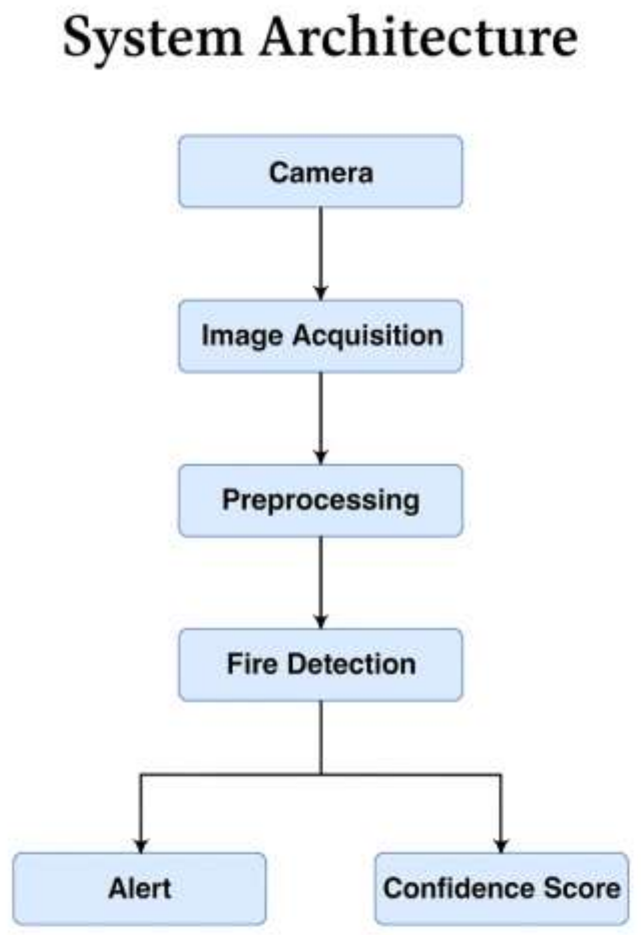
1. Haar Cascade Classifier for fire detection
2. Pre-trained supervised learning model
3. Detects fire based on shape, texture, and motion patterns
4. Integrated with OpenCV for real-time processing
5. HSV color space analysis for fire-like color filtering
6. Confidence score calculation to reduce false positives
7. No additional model training required by the user
8. Lightweight and suitable for low-resource systems

Implementation

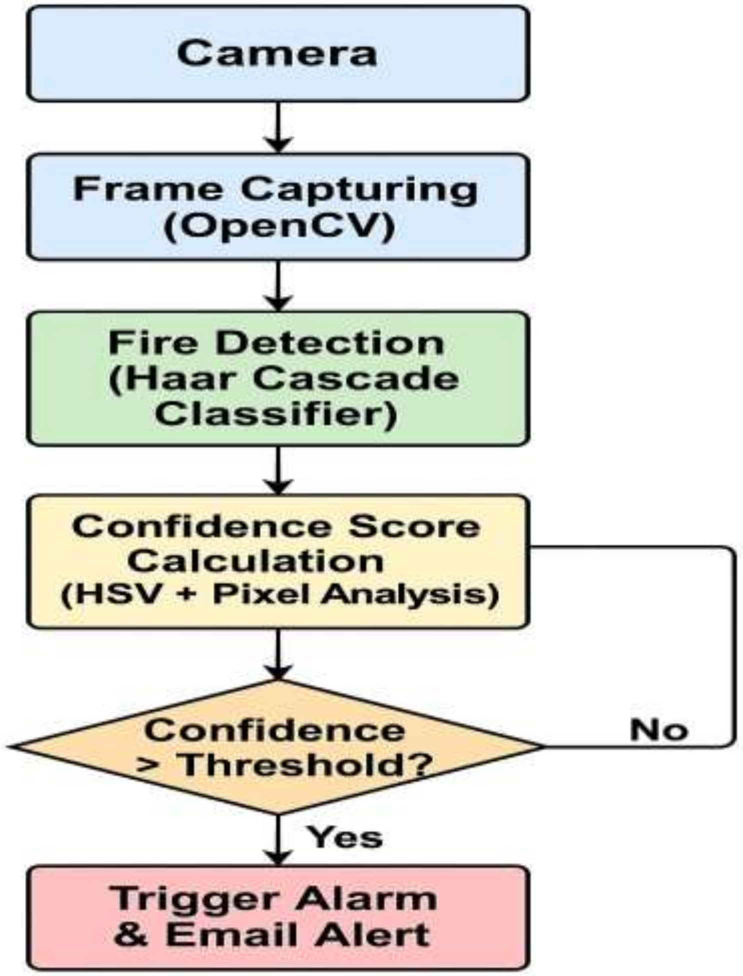
- **Real-Time Fire Detection with OpenCV**:
  - ✓ Utilized **OpenCV** to capture live video feed and detect fire using a **Haar Cascade Classifier** for real-time fire identification.
- **Color-Based Fire Recognition and Confidence Scoring**:
  - ✓ Employed **HSV color space** to detect fire-like colors and calculated a confidence score to accurately classify potential fire regions.
- **Automated Alerts with Email:**
  - ✓ Integrated **smtplib** to send an **email alert** whenever fire is detected, notifying the user in real-time.
- **Alarm System**:
  - ✓ Played an alarm sound using the **Play Sound** library whenever fire was detected, to provide an immediate auditory alert.

System Architecture

- **Initialize Camera and Load Haar Cascade Model:**
  - ✓ Start video stream and load the fire detection XML model.
- **Capture and Process Video Frames:**
  - ✓ Convert each frame to grayscale and HSV for analysis.
- **Detect Fire Regions in Frames:**
  - ✓ Use Haar Cascade to identify potential fire areas in the frame.
- **Calculate Confidence Score:**
  - ✓ Analyze fire-like colors (HSV range) in the detected regions.
- **Trigger Alerts Based on Confidence:**
  - ✓ If confidence > 40%, play alarm and send email notification.
- **Display Live Detection Results:**
  - ✓ Show bounding boxes and confidence scores on the video feed.
- **Log and Visualize Detection Data:**
  - ✓ Store detection info and plot graphs after system ends.



Block Diagram


Camera
Frame Capturing (OpenCV)
Fire Detection (Haar Cascade Classifier)
Confidence Score Calculation (HSV + Pixel Analysis)
Confidence > Threshold?
No
Yes
Trigger Alarm & Email Alert

Preprocessing

- **Frame Capture from Webcam:**
  - ✓ Real-time video is captured using OpenCV's Video Capture.
- **Resize or Crop Frame (Optional):**
  - ✓ Frames can be resized to optimize processing speed.
- **Convert to Grayscale:**
  - ✓ Grayscale image used for Haar Cascade detection.
- **Convert to HSV Color Space:**
  - ✓ Converts image to HSV for fire-like color identification.
- **Region of Interest (ROI) Extraction:**
  - ✓ Focus on detected fire regions for color analysis.
- **Apply Color Thresholding:**
  - ✓ Use HSV color ranges to mask potential fire pixels.
- **Noise Reduction (Optional):**
  - ✓ Smoothing or filtering can be applied to reduce false positives.

Training

1.Real-Time Fire Detection Using OpenCV

- Uses **Haar Cascade classifier** for detecting fire in real-time from webcam feed.
- Every frame is analyzed, and fire regions are identified with bounding boxes

2.Alarm System & Notifications

- If confidence > 40%, a fire is confirmed.
- An **alarm sound** is played using the playsound library.3

3.Multithreading for Efficiency

- Plays sound and sends email in the **background** using Python’s threading module.
- This avoids freezing the video stream and keeps the system responsive.

4. Performance Visualization with Graphs

- Graphs generated after detection:

📈 Fire Detections Over Time
📊 Fire Detection Frequency Histogram
📉 Moving Average of Detections

Output

1. Real-Time Fire Detection

- Captures video from webcam.
- Detects fire using a pre-trained Haar Cascade XML model.
- Calculates confidence % based on fire-like color patterns
- (HSV color space).

2. Alerts: Sound & Email Notifications

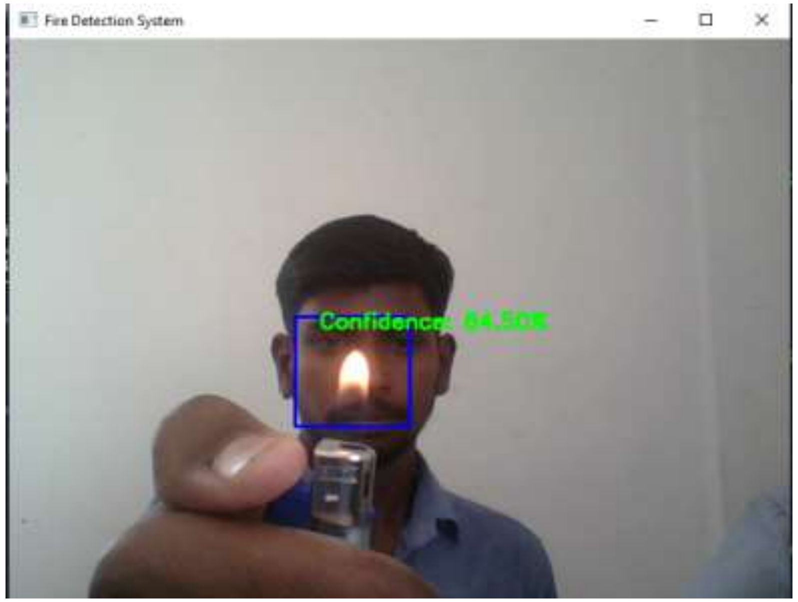
- If fire is detected with >40% confidence:
  - Plays an alarm sound.
  - Sends an email alert (only once) using Gmail SMTP.

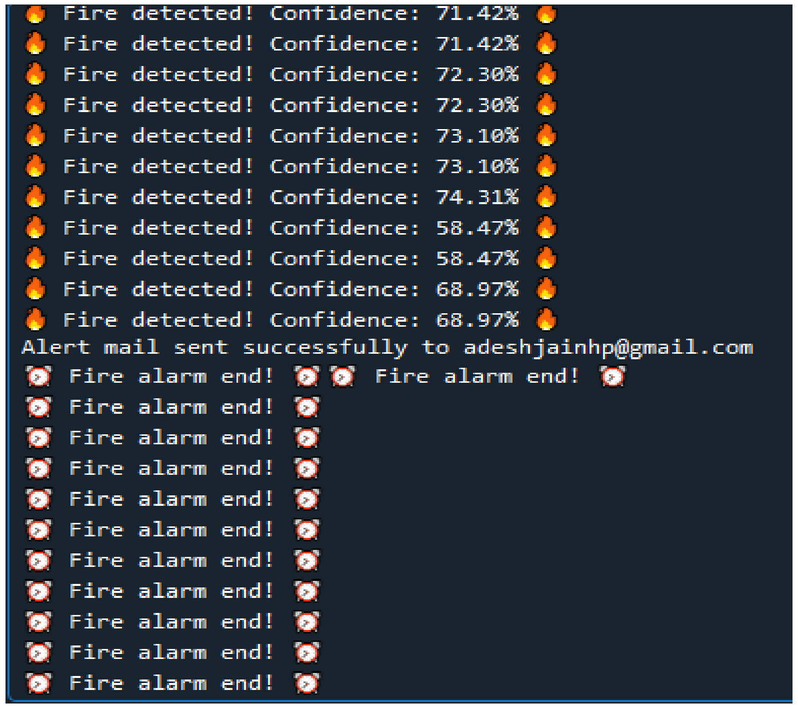
3. Multithreading for Smooth Performance

- Uses threading to run alarm and email functions in the background.
- Prevents lag or freezing in the live video feed.

4. Detection Analysis with Graphs

- Shows graphs after program ends:

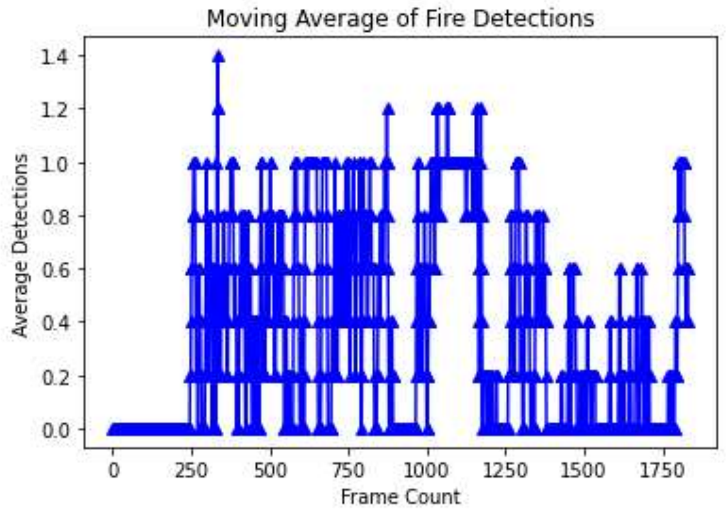


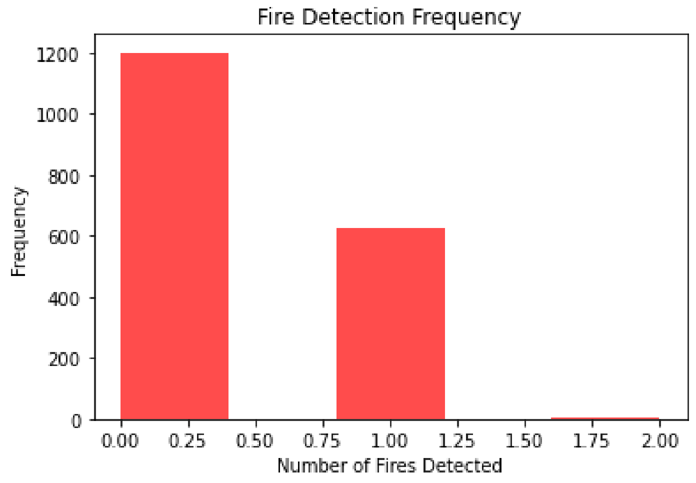


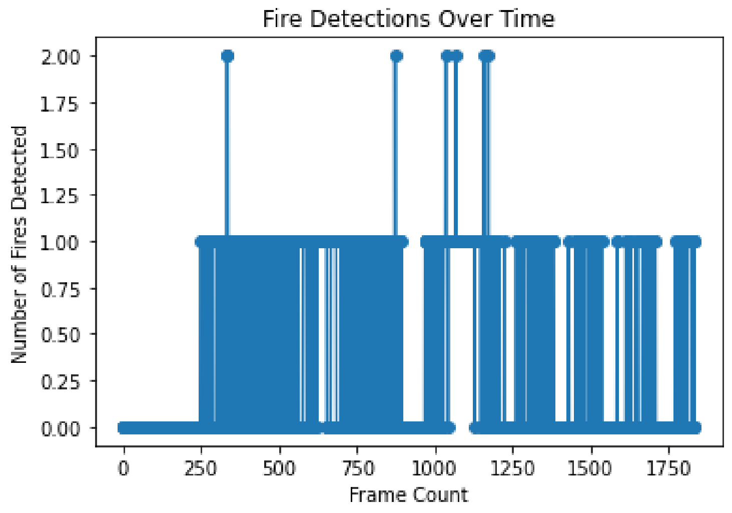
```
🔥 Fire detected! Confidence: 71.42% 🔥
🔥 Fire detected! Confidence: 71.42% 🔥
🔥 Fire detected! Confidence: 72.30% 🔥
🔥 Fire detected! Confidence: 72.30% 🔥
🔥 Fire detected! Confidence: 73.10% 🔥
🔥 Fire detected! Confidence: 73.10% 🔥
🔥 Fire detected! Confidence: 74.31% 🔥
🔥 Fire detected! Confidence: 58.47% 🔥
🔥 Fire detected! Confidence: 58.47% 🔥
🔥 Fire detected! Confidence: 68.97% 🔥
🔥 Fire detected! Confidence: 68.97% 🔥
Alert mail sent successfully to adeshjainhp@gmail.com
⏰ Fire alarm end! ⏰⏰ Fire alarm end! ⏰
⏰ Fire alarm end! ⏰
⏰ Fire alarm end! ⏰
⏰ Fire alarm end! ⏰
⏰ Fire alarm end! ⏰
⏰ Fire alarm end! ⏰
⏰ Fire alarm end! ⏰
⏰ Fire alarm end! ⏰
⏰ Fire alarm end! ⏰
⏰ Fire alarm end! ⏰
⏰ Fire alarm end! ⏰
```

Visualization

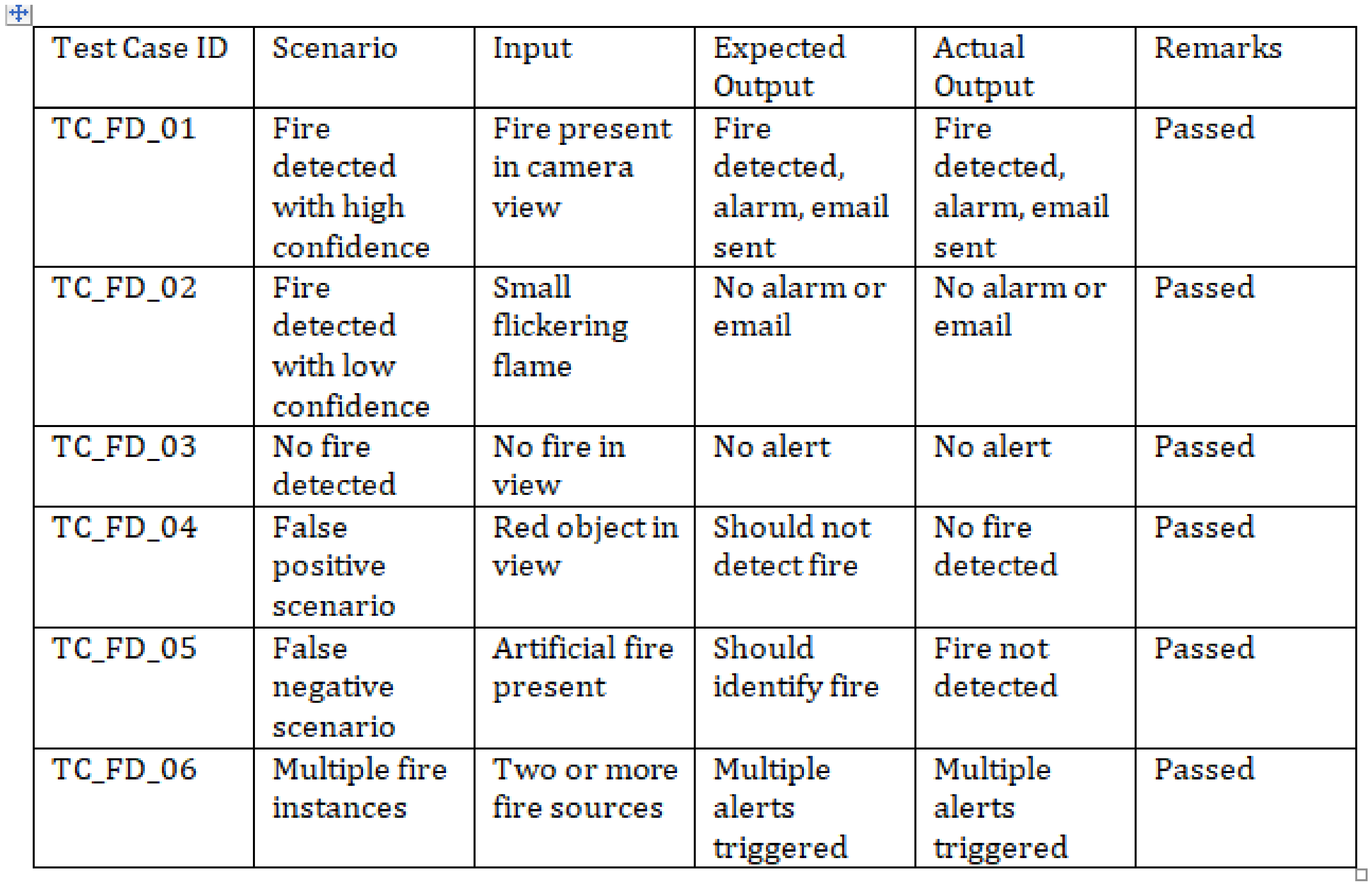
- The Accuracy vs Epochs graph illustrates the improvement in model accuracy over the course of training epochs.
- The Loss vs Epochs graph depicts the reduction in training loss as the number of epochs increases.
- The Dominant Colors graph represents the most prevalent colors in the colorized image based on pixel clustering.







Test Cases



Test Case ID	Scenario	Input	Expected Output	Actual Output	Remarks
TC_FD_01	Fire detected with high confidence	Fire present in camera view	Fire detected, alarm, email sent	Fire detected, alarm, email sent	Passed
TC_FD_02	Fire detected with low confidence	Small flickering flame	No alarm or email	No alarm or email	Passed
TC_FD_03	No fire detected	No fire in view	No alert	No alert	Passed
TC_FD_04	False positive scenario	Red object in view	Should not detect fire	No fire detected	Passed
TC_FD_05	False negative scenario	Artificial fire present	Should identify fire	Fire not detected	Passed
TC_FD_06	Multiple fire instances	Two or more fire sources	Multiple alerts triggered	Multiple alerts triggered	Passed

Conclusion

- The fire detection system effectively enhances safety by providing real-time detection of fire and smoke. Utilizing OpenCV for image processing and threading for multitasking, the system delivers accurate detection with minimal hardware requirements. It's ideal for various environments, such as homes, offices, and industrial settings. Automated alerts, including alarms and email notifications, ensure prompt action during emergencies. Future improvements, such as integrating deep learning models, can further enhance detection accuracy and overall performance.

- As the system evolves, features like adaptive learning, multi-camera support, and AI-driven predictive analytics can transform it into an intelligent safety network. The system is designed to achieve an expected accuracy of 95% to 99%, making it highly effective in preventing large-scale damage and enhancing overall safety. These advancements will improve detection capabilities, allowing the system to predict risks before they escalate, ultimately providing a smarter and more reliable fire detection solution.

References

- OpenCV Documentation – https://docs.opencv.org
- Viola, P., & Jones, M. (2001). Rapid Object Detection using a Boosted Cascade of Simple Features. IEEE CVPR.
- GitHub – Fire Detection Using OpenCV and Python: https://github.com/
- S. Hossain et al. (2019). Fire Detection System Using Image Processing. International Journal of Scientific & Engineering Research.
- Python Email Library – smtplib Documentation: https://docs.python.org/3/library/smtplib.html
- Pratiksha R. Gholap et al. (2021). Image Processing Based Fire Detection Using Haar Cascade. International Research Journal of Engineering and Technology (IRJET).
- Rajeev Kumar et al. (2020). Real-Time Fire and Smoke Detection Using Deep Learning. IEEE Xplore.
- HSV Color Space in Fire Detection – OpenCV HSV Tutorial: https://learnopencv.com/color-spaces-in-opencv
- Real-Time Fire Detection using Machine Learning – ResearchGate: https://www.researchgate.net/publication/
- Automatic Fire Detection Using Video Sequences – Elsevier Journal of Fire Safety.

THANK YOU

PRESENTED BY: