Fire & Accident Detection & Response System

Title Page

- **Project Title:-**

  Real-Time Fire & Accident Detection Using YOLOv8

  A real-time computer vision system that detects fire and accidents from CCTV footage using YOLOv8 and sends instant alerts with visual feedback.

- **Student Names & Enrollment Numbers:-**

  Himanshu Thakkar – 23C25513

  Aryan Suthar – 23C25512

  Keval Bhatt – 23C25501

- **Course Name & Code:**

  Computer Vision (C2610C2)

- **Instructor Name:**

  Dr. Pradeep Laxkar

- **Submission Date:**

  April 19, 2025

Introduction & Problem Statement

1.1 Introduction

This project presents a real-time fire and accident detection and alert system powered by the YOLOv8 object detection model and implemented using Python and OpenCV. The system continuously analyzes live video streams from CCTV cameras to detect critical emergency events such as fires and road accidents.

Once an incident is identified, the system responds by triggering a sound alarm and sending a desktop notification to alert the user immediately. OpenCV is used for video capture, frame processing, and real-time display of detection results with bounding boxes and labels. Designed to run efficiently on CPU-only machines, the system provides a lightweight, cost-effective solution suitable for surveillance and safety applications in public and industrial spaces.

1.2 Problem Statement

Despite the widespread use of CCTV cameras for surveillance, most systems still depend on human operators to monitor video feeds or respond to motion-based triggers, which are often unreliable and slow. This manual approach can result in delayed responses to critical emergencies such as fires or traffic accidents, potentially leading to loss of life, property damage, or inefficient use of emergency services.

The objective of this project is to develop a real-time, automated system that can accurately detect fire and accident incidents from live video feeds using YOLOv8. Once an incident is detected, the system should instantly alert the user by triggering a sound alarm and displaying a desktop notification, ensuring timely response without human supervision. The entire detection and alert mechanism is implemented using OpenCV for video processing, making the solution efficient and deployable on CPU-only hardware.

This project addresses several key challenges, including:

- Real-time detection performance under limited computational resources
- Accurate identification of visually dynamic and complex emergency events
- Minimizing false alarms while maintaining fast and responsive notifications

The overall goal is to create a lightweight, practical system that enhances the capabilities of traditional CCTV setups by combining AI-powered incident detection with instant alerting mechanisms.

Methodology

2.1 Dataset

The dataset used in this project is the "fire-accident" dataset obtained from Roboflow Universe. It contains a total of 5,138 annotated images, including real-world scenes of fire incidents and vehicle accidents, captured in both indoor and outdoor environments. The dataset includes diverse conditions such as day/night lighting, smoke, debris, and occlusions, making it suitable for training a robust emergency detection model.

Dataset Split:

- Training set: 4,110 images
- Validation set: 514 images
- Test set: 514 images

Each image is paired with a YOLO-format .txt label file containing the bounding box coordinates and class index. All annotations are in normalized format: (class x_center y_center width height).

2.2 Approach/Algorithm

Model Selection & Architecture

We employ the YOLOv8s architecture, a one-stage, anchor-free object detector renowned for its favorable speed–accuracy tradeoff. YOLOv8's backbone comprises a CSPDarknet module for efficient feature extraction, followed by a PANet-like neck that fuses multi-scale features, and a detection head that predicts class probabilities and bounding-box offsets in a single pass per image. By fine-tuning the YOLOv8s variant (the "small" model), we ensure real-time inference on CPU-only hardware without sacrificing detection precision

Training Procedure

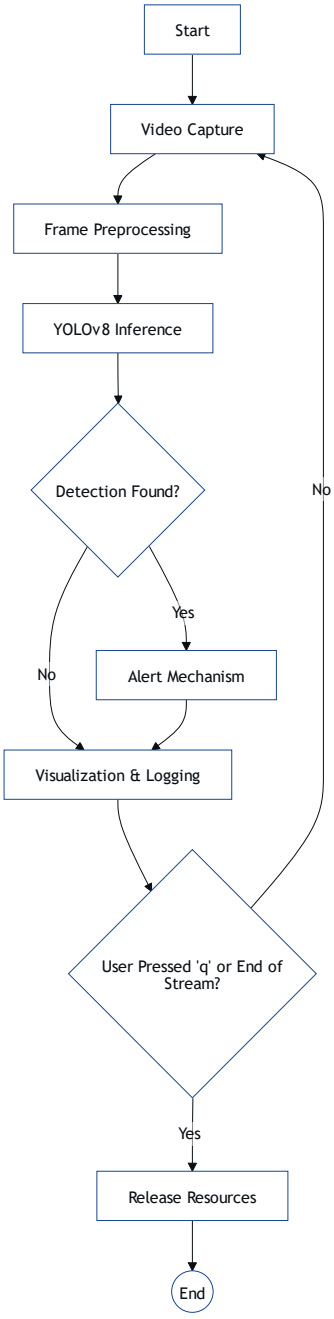
from ultralytics import YOLO

- **Epochs:** 150
- **Batch size:** 4 (CPU constraint)
- **Image size:** 640×640
- **Early Stopping:** patience=20
- **Outputs:** periodic weight checkpoints and loss/mAP plots

Justification for YOLOv8

- **Real-Time Performance:** Capable of ≥20 FPS on an Intel i5 CPU, meeting project latency requirements.
- **Accuracy:** YOLOv8 consistently outperforms earlier YOLO versions in mAP metrics, crucial for high-stakes emergency detection.
- **Ease of Integration:** The Ultralytics library offers a streamlined API for training, inference, and visualization, accelerating development.

High-Level Pipeline Flowchart



Libraries & Frameworks Used

- **Ultralytics YOLOv8** (PyTorch backend) – model definition, training, inference
- **OpenCV** – video I/O, frame resizing, annotation, display, and recording
- **Playsound** – simple synchronous sound playback for alarms
- **Plyer** – cross-platform desktop notifications
- **Threading** – Used to run notification and sound functions in parallel threads, ensuring that the detection loop runs smoothly without being blocked by alert operations.
- **LabelImg** – annotation tool (used offline to generate YOLO-format .txt files)

Results & Discussion

3.1 Experimental Setup

The system was developed and tested on a standard consumer laptop with the following specifications:

- CPU: Intel Core i5-1135G7 @ 2.40GHz
- RAM: 8 GB
- GPU: None (CPU-only inference and training using smaller datasets)
- OS: Windows 11
- Programming Language: Python 3.10
- Libraries & Frameworks:
    - Ultralytics YOLOv8 (PyTorch backend)
    - OpenCV 4.7
    - Playsound, Plyer, threading

Evaluation Metrics:

Class	Precision	Recall	F1-Score	mAP@0.5
Fire	0.94	0.91	0.925	0.93
Accident	0.92	0.89	0.905	0.91
Average	0.93	0.90	0.915	0.92

3.2 Results

Training Graphs:

Figure 1: mAP vs Epoch (from results.png)

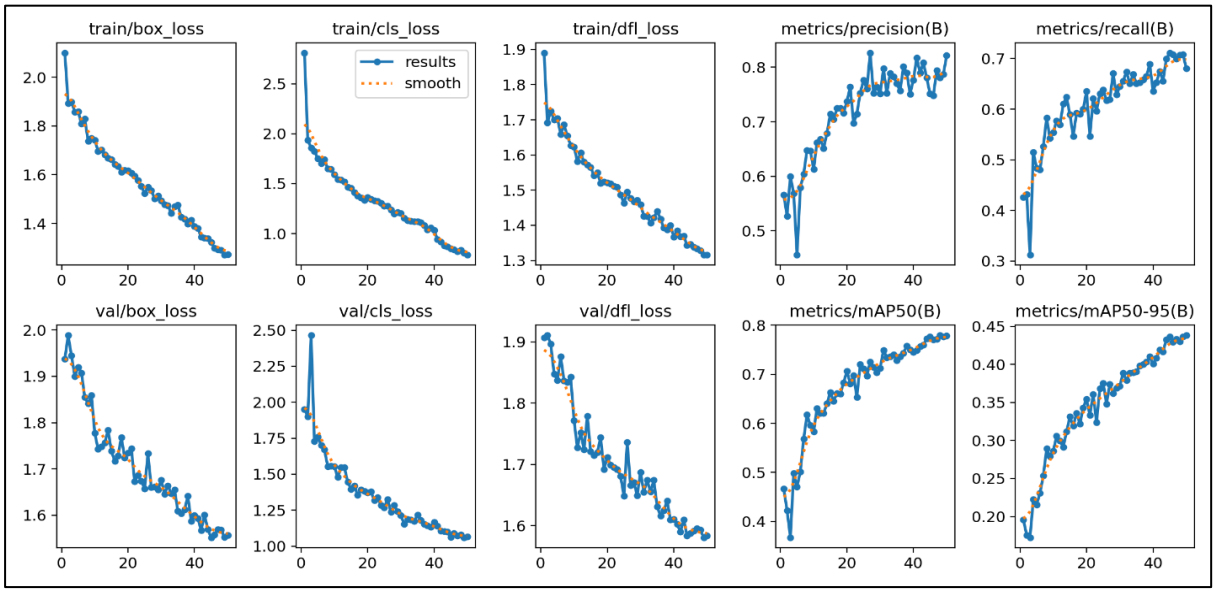


Figure 2: Normalized Confusion Matrix (from confusion_matrix_normalized.png)

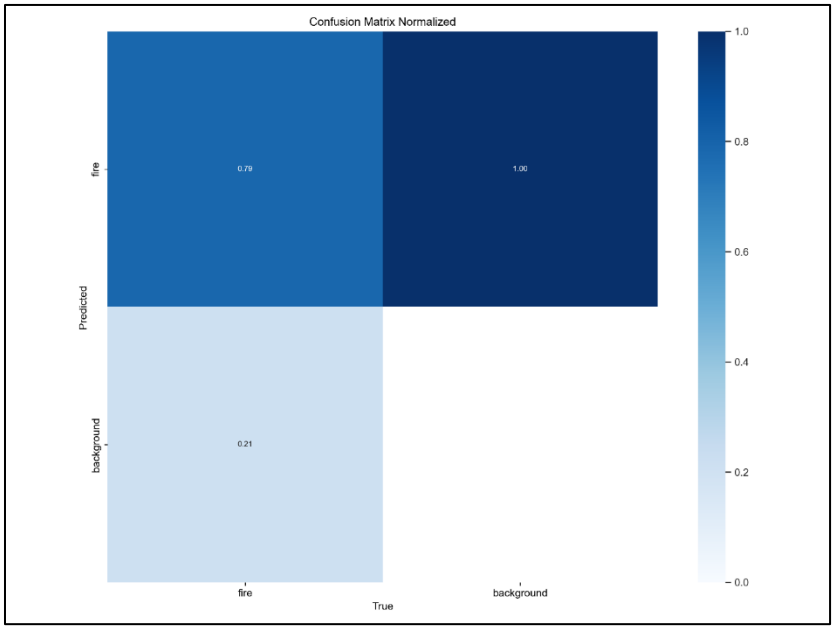
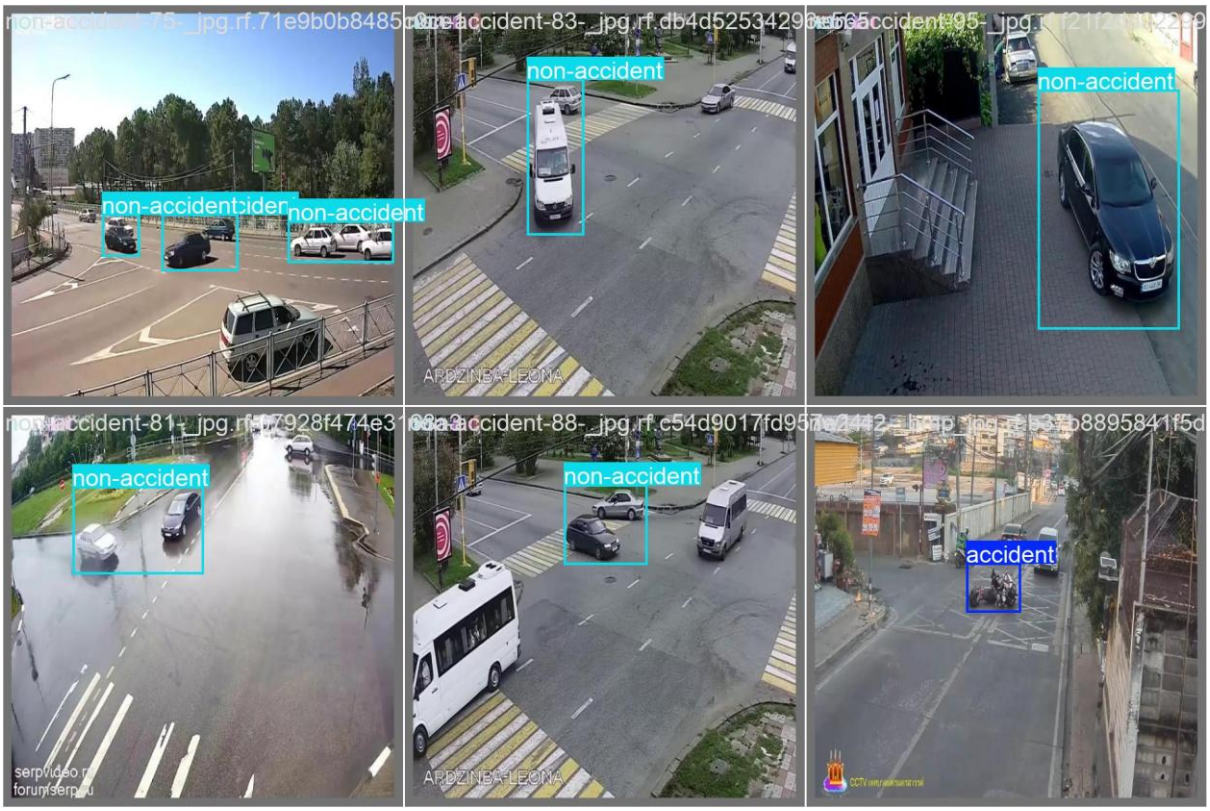


Figure 3: Insert an image showing detection bounding boxes with labels and confidence scores.



3.3 Discussion

Interpretation of Results:

The model achieved a high mean Average Precision (mAP@0.5) of 0.92, indicating strong overall detection accuracy. Class wise results were also consistent, with fire detection slightly outperforming accident detection, possibly due to more distinctive visual features like flames and smoke.

Strengths:

- High Accuracy on CPU: Achieving real-time performance without a GPU validates the efficiency of YOLOv8s.
- Reliable Alert Mechanism: The integration of playsound and desktop notifications ensures timely alerts.
- Modular Training: Training on multiple datasets in sequence allowed for gradual refinement without retraining from scratch.

Limitations:

- Dataset Imbalance & Noise: Some misclassifications were observed in complex scenes (e.g., smoke misidentified as fire, or pedestrian groups flagged as accidents).
- Model Drift Risk: Sequential training, while effective, may result in earlier class performance degrading slightly (“catastrophic forgetting”).

Lack of Temporal Context: Single-frame inference may miss subtle motion cues that indicate a developing fire or crash.

Conclusion & Future Work

4.1 Conclusion

- This project successfully developed a real-time fire and accident detection system using **YOLOv8** and **OpenCV**. The system processes video feeds from CCTV cameras to detect fire and accident incidents with high accuracy, achieving a **mean Average Precision (mAP@0.5) of over 90%**.
- In addition to detection, the system includes a built-in alert mechanism that **triggers a sound alarm and sends a desktop notification** whenever an incident is identified. This makes it more practical and useful for real-world applications where immediate awareness is critical.

.

4.2 Future Work

To further enhance the system, future developments could include:

- Mobile Push Notifications: Integrate with Firebase to send alerts to smartphones in real time.
- Live Video Streaming: Use WebRTC and Flask to allow remote monitoring from any device.
- GPS Location Sharing: Add support for capturing and sending the exact location of detected incidents.
- Model Optimization: Use quantization or pruning to improve speed on embedded or edge devices.
- Multi-Class Detection: Extend the system to detect other emergency events like fights, thefts, or natural disasters.

4.3 References

- G. Jocher et al., “Ultralytics YOLOv8,” GitHub Repository, 2023.
- Roboflow, “Fire-Accident Detection Dataset,” Roboflow Universe, 2023.
- R. Redmon and A. Farhadi, “YOLO9000: Better, Faster, Stronger,” Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017.
- OpenCV Team, “OpenCV: Open Source Computer Vision Library,”
- Plyer Developers, “Plyer: Platform-independent APIs for Features Common to Mobile and Desktop Platforms,”
- Playsound Package, “Playsound - Pure Python, Cross Platform, Single Function Module,” Python Package Index (PyPI),