Real-Time Smoke and Fire Detection with Automated SMS Alerting Using CCTV Surveillance Footage

Akshat Parmar, Tharun Harish, Vikranth Udandarao, Vimal Jayant Subburaj

Group number 9

CV Project Interim Presentation

INDRAPRASTHA INSTITUTE *of* INFORMATION TECHNOLOGY DELHI

Problem statement, scope & users

Problem Statement:

- Rising fire incidents in urban & industrial areas threaten lives and property.
- Delayed detection increases damage; traditional systems lack real-time response.
- Need robust, automated smoke & fire detection for CCTV networks.

Scope:

- Develop a web app for real-time smoke & fire detection using YOLOv8n.
- Train on D-Fire Dataset (14,122 train, 3,099 val images) for baseline performance.
- Enable user uploads of live CCTV feeds & automated alerts with GPS.
- Current phase: Baseline model (mAP@0.5: 0.743, mAP@0.5:0.95: 0.426).
- Future: Enhance accuracy, scale to multiple feeds, deploy alerting system.

Users:

- **Primary**: Building managers, security personnel monitoring CCTV systems.
- **Secondary**: Fire stations receiving automated alerts with location data.
- **End Users**: Residents, workers benefiting from early fire detection.

Related work

- **Traditional Approaches:**
  - Haar cascades & color-based methods: Fast but struggle with smoke variability (lighting, density).
  - Thresholding (e.g., RGB/HSV): Limited by false positives in complex scenes.

- **Early Deep Learning:**
  - CNNs (e.g., AlexNet, VGG): Image classification for fire/smoke; high accuracy, low speed.
  - Challenges: Not suited for real-time CCTV due to computational cost.

- **Modern YOLO-Based:**
  - YOLOv3: mAP ~0.7, faster but less precise for smoke (Li et al., 2019).
  - YOLOv5: mAP ~0.75–0.8, balances speed/accuracy (Khan et al., 2021).

- **Datasets:**
  - FireNet: Small-scale, controlled fire/smoke images.
  - D-Fire: Larger, diverse real-world scenarios (used in this work).

Baseline methods

- **Model Selection:**
  - YOLOv8n (nano): 3M params, 8.2 GFLOPs; lightweight for real-time detection.

- **Training Setup:**
  - Dataset: D-Fire (14,122 train, 3,099 val images).
  - Hardware: NVIDIA RTX 4060 GPU, AMP enabled.
  - Config: 100 epochs, batch 16, 640x640 imgs.

- **Hyperparameters:**
  - Optimizer: SGD (lr=0.01, momentum=0.9).

- **Performance:**
  - mAP@0.5: 0.743, mAP@0.5:0.95: 0.426.
  - Class-Specific: Smoke: 0.497, Fire: 0.356.

- **Key Observations:**
  - Smoke detection stronger than fire; baseline sets optimization target.

Dataset & Evaluation Metrics

Dataset:

- **Source:** D-Fire Dataset.
- **Training Set:** 14,122 images (6,458 without objects).
- **Validation Set:** 3,099 images (1,375 without objects).
- **Classes:** Smoke, Fire; real-world CCTV scenarios.
- **Preprocessing:** Corrupt JPEGs restored.

Evaluation Metrics:

- **Precision:** 0.758 (overall detection confidence).
- **Recall:** 0.672 (proportion of true positives detected).
- **mAP@0.5:** 0.743 (accuracy at IoU=0.5).
- **mAP@0.5:0.95:** 0.426 overall (smoke: 0.497, fire: 0.356).
- **Observation:** Smoke outperforms fire; reflects dataset bias or visual cues.

System

Overview:

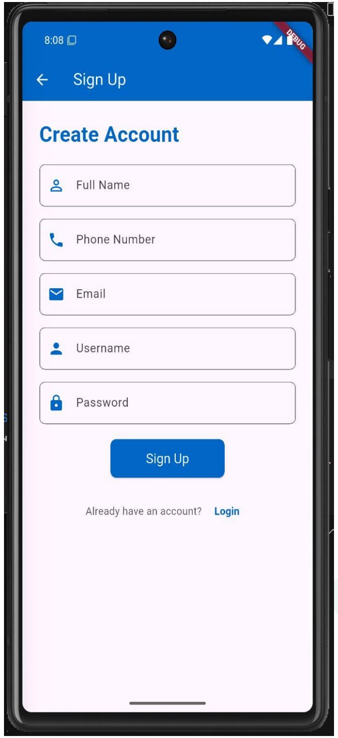
- Mobile app for real-time smoke & fire detection.
- **Frontend:** User login, live CCTV feed uploads.
- **Backend:** YOLOv8n model processes feeds.

Functionality:

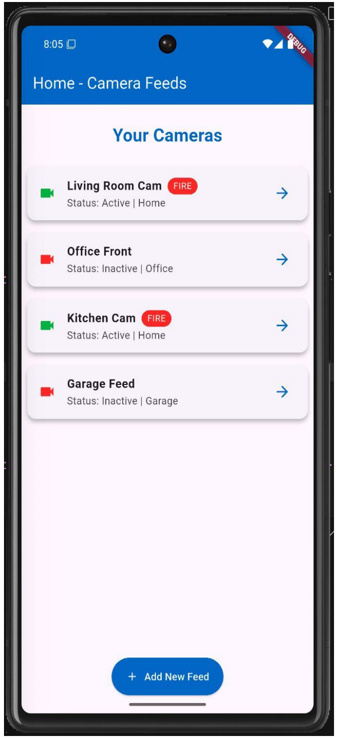
- Detects smoke/fire in live streams.
- Sends alerts: App notifications + fire station API (GPS included).

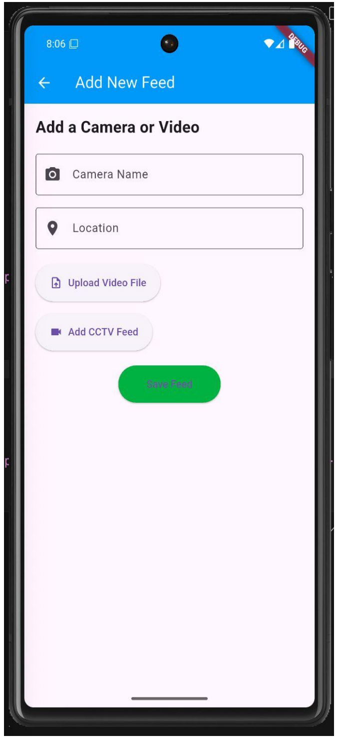
Tech Stack:

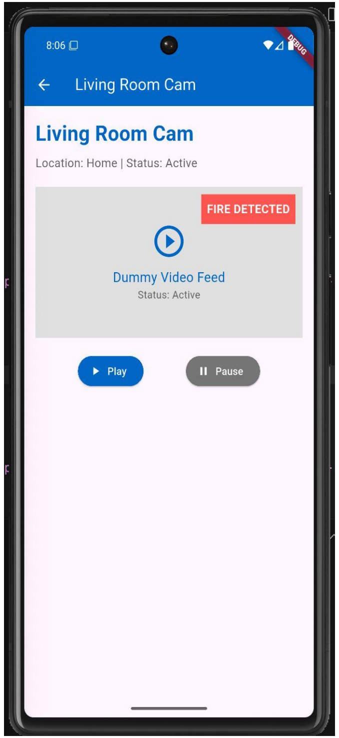
- **Frontend:** Flutter (Dart)
- **Backend:** Flask (Python)
- **Database:** MongoDB
- **AI/ML:** YOLO8n

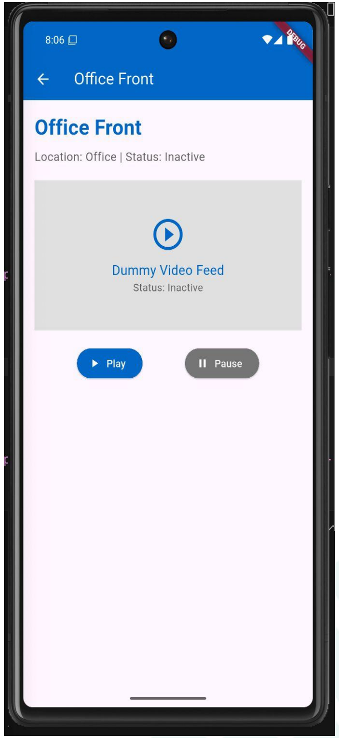


System


8:05
Home - Camera Feeds
Your Cameras
Living Room Cam FIRE
Status: Active | Home
Office Front
Status: Inactive | Office
Kitchen Cam FIRE
Status: Active | Home
Garage Feed
Status: Inactive | Garage
+ Add New Feed


8:06
Add New Feed
Add a Camera or Video
Camera Name
Location
Upload Video File
Add CCTV Feed
Save Feed


8:06
Living Room Cam
Living Room Cam
Location: Home | Status: Active
FIRE DETECTED
Dummy Video Feed
Status: Active
Play
Pause


8:06
Office Front
Office Front
Location: Office | Status: Inactive
Dummy Video Feed
Status: Inactive
Play
Pause

Baseline results

Model Name	Precision	Recall	mAP@0.5	mAP@0.5:0.95	Compute Time (s)
YOLOv8n	0.758	0.672	0.743	0.426	15.2
YOLOv8s	0.772	0.689	0.759	0.439	18.1
YOLOv8m	0.785	0.704	0.771	0.452	21.3

Next Steps

- **Improve Fire Detection:** Address lower fire mAP@0.5:0.95 (0.356 vs. 0.497 for smoke) by augmenting fire data and fine-tuning YOLOv8 models.
- **Reduce Background Errors:** Mitigate misclassifications (e.g., 405 smoke-to-background errors) using advanced background subtraction techniques.
- **Real-Time Validation:** Test YOLOv8n on live CCTV feeds to ensure compute time (15.2 s on test video) supports real-time mobile app deployment.
- **Model Optimization:** Explore quantization and pruning to reduce latency while maintaining accuracy.

Individual Contributions

Akshat Parmar: Fine-tuned YOLOv8n base model, focusing on hyperparameter optimization and training pipeline setup.

Tharun Harish: Developing mobile app, integrating real-time fire/smoke detection with YOLOv8n inference.

Vikranth Udandarao: Conducted literature review and analyzed related work for base model research.

Vimal Subburaj: Prepared D-Fire Dataset, handled data preprocessing, and evaluated model performance.

Feedback & Next Steps

Feedback:

- **Perform in-depth dataset analysis to assess data distribution, identify biases, and ensure adequate representation across classes.**
- **Conduct thorough error analysis** to understand where and why the model is underperforming, and identify common failure patterns.

Next Steps:

- **Develop more localized datasets** to improve model relevance and performance in specific regions or contexts.
- **Enhance the Mean Average Precision (MaP) score** through model optimization and better training strategies.
- **Implement robust person detection algorithms** to accurately identify individuals in the given context or environment.
- **Complete the remaining stages of app development**, including UI refinement, backend integration, and performance optimization.

Thank You