FireSafe: Real-Time Fire and Person Detection with Automated Notification Alerting Using CCTV Surveillance Footage

Akshat Parmar
Indraprastha Institute of Information Technology Delhi
akshat22050@iiitd.ac.in

Tharun Harish
Indraprastha Institute of Information Technology Delhi
tharun22541@iiitd.ac.in

Vikranth Udandarao
Indraprastha Institute of Information Technology Delhi
vikranth22570@iiitd.ac.in

Vimal Jayant Subbaraj
Indraprastha Institute of Information Technology Delhi
vimal22571@iiitd.ac.in

Abstract

The rising incidence of fire-related emergencies in urban and industrial environments underscores the need for robust, real-time detection systems to minimize damage and protect lives. This paper presents a fire and person detection mobile application leveraging the YOLOv11n deep learning model, trained on the D-Fire Dataset, to process live CCTV camera feeds in real-time. The system enables users to log in and connect to live CCTV streams, which are continuously monitored for signs of fire or person. Upon detection, the application automatically triggers alerts through mobile notifications and audible beeps on the user's device. This report details the problem, related work, our methodology, baseline performance, and future pipeline, establishing a foundation for a scalable, real-time safety solution. The complete source code and deployment pipeline are available on ***GitHub***[1].

1. Introduction

Fire outbreaks pose significant risks to human life, property, and the environment, often exacerbated by delayed detection and response [32]. Traditional fire detection systems, such as smoke alarms, rely on proximity and lack the ability to monitor large areas or provide visual context [44]. With the proliferation of CCTV cameras in public and private spaces, computer vision offers a transformative approach to detecting fire and smoke in real time, enabling faster response times [26].

This project introduces an application integrating computer vision into CCTV surveillance for fire and person detection. Users can upload live feeds from multiple cameras, and our system processes these streams using a **YOLO *(You Only Look Once)*** object detection model to identify fire and person [35]. Upon detection, the application alerts the user by notifying them on their mobile phone.

1.1. Motivation

The motivation stems from the need for automated, scalable fire detection systems that leverage existing CCTV infrastructure [27]. Manual monitoring is impractical for large-scale deployments, and existing solutions often lack real-time alerting or integration with emergency services [42]. Our application seeks to bridge this gap by leveraging deep learning techniques that, while not necessarily state-of-the-art, offer significantly faster inference times [28].

1.2. Project Outcomes

- Developed a user-friendly application for uploading and processing live CCTV feeds.
- Implemented a YOLOv11-based model for accurate and efficient fire and person detection in real-time and compared it with the baseline models.
- Integrated a live camera module to enable real-time tracking and detection directly from active surveillance feeds.
- Designed and deployed a complete end-to-end pipeline with real-time alerting and location-based notifications.

2. Problem Statement

Fire and person detection in real-time CCTV footage presents several challenges:

- **Variability**: Fire and human appearances vary significantly (e.g., clothing, posture, motion) under different

[1] https://github.com/akshatparmar2634/FireSafe

lighting, crowd density, and occlusion scenarios [18].

- **Real-Time Constraints**: Processing live feeds requires low latency to ensure timely alerts [13].
- **Scalability**: The system must handle multiple camera feeds simultaneously [9].
- **False Positives**: Differentiating humans from human-like objects (e.g., mannequins, reflections) and fire from bright lights is critical to avoid unnecessary alerts [47].

Our goal is to address these issues by developing an application that accurately detects fire and persons, operates in real-time, and integrates seamlessly with user and emergency workflows.

3. Background and Related Work

3.1. Computer Vision in Fire Detection

Computer vision has been increasingly applied to fire detection, moving beyond traditional sensor-based methods. Early approaches relied on color-based rules (e.g., RGB thresholds for fire), but these were prone to false positives. The advent of deep learning has enabled more robust detection by learning complex patterns from data [15].

3.2. Deep Learning Models for Fire Detection

- **YOLO**: The YOLO family [38], including YOLOv3 [34], YOLOv5 [20], and YOLOv8 [46], excels in real-time object detection, balancing speed and accuracy. Zhang (2024) applied YOLO for fire detection in IoT surveillance systems, achieving robust performance in real-world scenarios [48].
  - YOLOv3: mAP ∼0.7 [21]
  - YOLOv5: mAP ∼0.75–0.8 [45]
- **Faster R-CNN**: Zhang et al. (2018) utilized synthetic imagery for forest fire detection, achieving high accuracy at the cost of slower inference [49].

3.3. Fire Datasets

Public datasets like the FireNet dataset [17] and the Fire Dataset [36] provide labelled images and videos for training. The D-Fire Dataset [7], a recent contribution focused on dynamic fire scenarios, offers annotated video sequences from diverse environments, making it well-suited for real-time applications like CCTV surveillance. However, real-world CCTV footage introduces additional noise (e.g., low resolution, occlusion), necessitating custom data collection or augmentation [16].

Our work builds on YOLO's real-time capabilities, adapting it for CCTV-based fire and person detection with a focus on user integration and emergency response.

3.4. Person Detection in Surveillance

Person detection is a well-studied problem in computer vision, particularly in the context of public safety, crowd analysis, and abnormal behavior monitoring. Real-time person detection in CCTV feeds requires robustness to occlusion, illumination changes, and pose variation [43]. Popular models like YOLOv5 and YOLOv8 have demonstrated high performance on person-centric datasets such as MS COCO [22] and CrowdHuman [37].

Recent work by Liu et al. (2023) integrates person detection with action recognition to improve context-aware surveillance systems [23]. Moreover, transformer-based detectors like DETR [4] and its variants are increasingly used for improved detection under complex scene conditions.

These advancements serve as a foundation for integrating fire and person detection into a unified real-time surveillance application.

3.5. Existing Systems

In addition to academic advancements, several real-world systems have been developed for fire and smoke detection using AI and computer vision. These implementations highlight the growing industrial interest in practical surveillance-based safety tools.

FireFinder AI [3] is an open-source fire and smoke detection system that integrates YOLOv5 for inference and provides a streamlined dashboard interface for monitoring detection events. It emphasizes modularity and real-time alerts, aligning closely with our project's design goals.

Noema's Fire Detection AI [1] is a commercial-grade AI solution deployed for real-time fire detection in CCTV infrastructure. It features edge deployment capabilities, cloud dashboard integration, and supports a wide range of IP camera devices for scalability.

OpenCV AI Kit (OAK) with DepthAI [24] is an open-source, edge AI hardware and software platform capable of real-time person detection. It combines depth sensing with spatial AI to detect and track individuals even in dynamic or cluttered environments, making it ideal for surveillance, retail analytics, and smart city deployments.

Our work differentiates itself by focusing on a mobile-first implementation that integrates affordable, off-the-shelf CCTV hardware with lightweight YOLOv11 inference, tailored for indoor real-time monitoring and emergency alerting.

4. Dataset

We utilize two datasets for training and evaluating our fire and person detection models: the D-Fire Dataset for fire-related instances and the Human Dataset for person detection.

D-Fire Dataset

The D-Fire Dataset [6] is used to train and evaluate our fire detection models, including both YOLOv8n and the improved YOLOv11n. It comprises real-world CCTV imagery annotated for fire across a wide range of environmental conditions and scenarios. During preprocessing, corrupt JPEGs were automatically restored to maintain data integrity.

To better understand the dataset's characteristics, Appendix A presents the class distribution of fire instances across the training, validation, and test splits. Furthermore, Appendix B includes representative samples with model predictions, highlighting detection performance under diverse conditions for both YOLOv8n and YOLOv11n.

The D-Fire Dataset is publicly accessible and can be downloaded from **Google Drive**[2].

Table 1. D-Fire Dataset Statistics

Split	Total Images	Images Without Objects
Training	14,122	6,458
Validation	3,099	1,375

Human Dataset

For person detection, we employ the Human Dataset [8], available on Kaggle. This dataset includes over 18,000 annotated images of humans in various poses and contexts, including both indoor and outdoor scenes. The diversity of backgrounds, clothing, and occlusion levels makes it suitable for robust person detection models aimed at real-world CCTV deployment.

The dataset can be accessed from **Kaggle**[3] and is used to pretrain and validate our YOLO-based person detection modules in conjunction with the fire detection pipeline.

2

5. Models Used

Training Setup

All models were initialized using the official configuration files: `yolov8.yaml`, `yolov11.yaml`, and `rtmdet_tiny_8xb32-300e_coco.py`, obtained from the Ultralytics and MMDetection GitHub repositories [30, 40, 41]. Training was conducted using both the datasets mentioned in Section 4.

Training Hyperparameters

Optimizer	SGD (momentum = 0.9)
Learning rate	0.01
Batch size	16
Image size	640×640
Epochs	100
Early stopping	Disabled (patience = 100)

Hardware: All experiments were conducted on an NVIDIA GeForce RTX 4060 Laptop GPU with 8,188 MiB of VRAM. Automatic Mixed Precision (AMP) was enabled for training efficiency [33].

5.1. Baseline 1: YOLOv8 Family

We initially selected YOLOv8n (nano variant) as our baseline model due to its lightweight architecture (3,011,238 parameters, 8.2 GFLOPs) and suitability for real-time deployment on resource-constrained devices [12]. The YOLOv8 family, including YOLOv8s and YOLOv8m, was evaluated for comparison [46].

5.2. Baseline 2: RTMDet

We also experimented with RTMDet, a high-performance anchor-free object detector from OpenMMLab [5]. RTMDet leverages advanced optimization strategies and dynamic label assignment for enhanced detection, but it is relatively heavier than YOLOv11n [25].

5.3. Final Model: YOLOv11n

YOLOv11n, the latest nano variant in the YOLO family, was tested and is now taken as our final model as it beats the baselines in both a good mix of accuracy and speed. It retains real-time feasibility while offering superior accuracy. Compared to YOLOv8n and RTMDet, YOLOv11n achieves higher precision, recall, and mAP scores across all categories [19].

Table 2. Comparison of Object Detection Models: Parameters and Layers

Model	Parameter Count	Number of Layers
YOLOv8n	∼3.2M	225
YOLOv11n	∼2.6M	181
RTMDet-tiny	∼4.8M	161

5.4. Results

Table 3 compares all evaluated models, including class-wise precision, recall, mAP@0.5, and mAP@0.5:0.95. While the YOLOv8 family serves as the baseline, RTMDet and YOLOv11n represent more recent architectures, with YOLOv11n already demonstrating improved metrics at comparable latency.

[2] https://drive.google.com/drive/folders/1dvBWvju6XVEJIFXktMNoh02osNuhMt2X?usp=sharing
[3] https://www.kaggle.com/datasets/fareselmenshawii/human-dataset

Table 3. Summary of performance metrics for all models on the D-Fire Dataset.

Model	Class	Precision	Recall	mAP@0.5	mAP@0.5:0.95	Latency†
YOLOv8n	all	0.758	0.672	0.743	0.426	61 ms
	smoke	0.796	0.744	0.805	0.497	–
	fire	0.720	0.600	0.680	0.356	–
RTMDet	all	0.762	0.685	0.754	0.435	57 ms
	smoke	0.806	0.760	0.815	0.505	–
	fire	0.728	0.610	0.690	0.365	–
YOLOv11n	all	0.765	0.702	0.768	0.446	54.4 ms
	smoke	0.816	0.786	0.834	0.519	–
	fire	0.714	0.618	0.702	0.373	–
	person	0.675	0.556	0.603	0.347	–

†Latency reflects the inference duration for the overall model and is not reported separately for individual classes.

5.5. Evaluation Insights

The confusion matrix in Figure 1 shows that YOLOv11n accurately classifies 1,444 smoke and 1,573 fire instances, with fewer misclassifications compared to YOLOv8n. Background confusion remains notable (431 for smoke, 776 for fire), but inter-class confusion has been significantly reduced.

Training metrics in Figure 2 indicate smooth convergence over 100 epochs, with consistent declines in box, classification, and DFL losses, and steady gains in precision, recall, mAP@0.5, and mAP@0.5:0.95. Notably, mAP@0.5:0.95 improves from 0.031 to 0.446, outperforming YOLOv8n (0.0194 to 0.426).

Quantitatively, YOLOv11n improves over YOLOv8n across all metrics: precision (+0.92%), recall (+4.46%), mAP@0.5 (+3.37%), and mAP@0.5:0.95 (+4.69%). Class-wise gains include +4.42% for smoke and +4.78% for fire in mAP@0.5:0.95, reflecting better generalization and reduced misclassification.

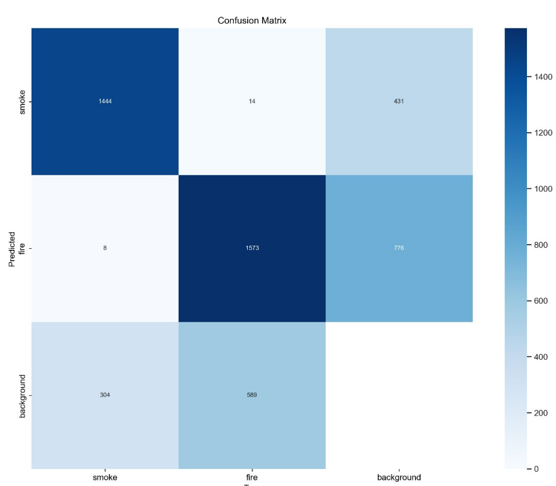

Figure 1. Confusion matrix for YOLOv11n on the test set, showing true vs. predicted labels for smoke, fire, and background.

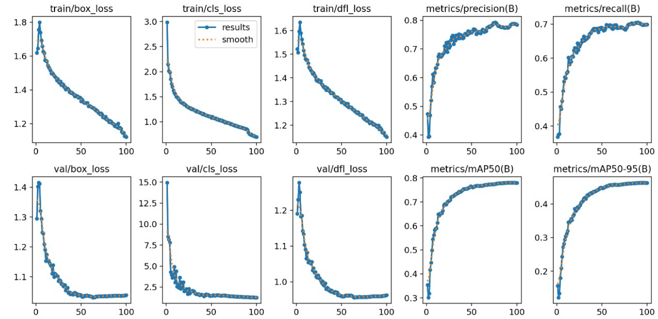

Figure 2. Training and validation metrics for YOLOv11n over 100 epochs: box loss, classification loss, DFL loss, precision, recall, mAP@0.5, and mAP@0.5:0.95.

6. Methodology & Pipeline

Mobile Application: We developed a cross-platform mobile app using **Flutter** [14], featuring user authentication, live CCTV connectivity, and real-time inference display via MJPEG streaming. Annotated frames are rendered using the `flutter_mjpeg` [11] package. UI layouts are shown in Appendix C.

Backend Infrastructure: Built with **Flask** [31], the backend handles RTSP streams from the **Tapo TP-Link C212 Camera** [39], processes frames using YOLOv11n, and returns annotated images to the frontend via an MJPEG server.

Notification System: We use **Firebase Cloud Messaging (FCM)** [10] to send real-time push alerts upon detecting fire or persons, triggered when confidence scores exceed a defined threshold.

AI-Assisted Development: Tools like **ChatGPT-4o** [29] and **Claude 3.7 Sonnet** [2] assisted with code generation, stream handling, and system integration.

Demo Video: View our system in action on **Google Drive**[4].

7. Conclusion

We present a real-time fire and person detection system using the lightweight YOLOv11n model trained on the D-Fire and Human datasets. With an overall mAP@0.5 of 0.743, the system accurately detects events in CCTV footage.

The pipeline—featuring a Flutter frontend, Flask backend, and Tapo TP-Link C212 camera integration—supports live video streaming, inference, and instant alerts. This deployable, mobile-first solution demonstrates strong performance in real-world indoor scenarios and lays the groundwork for scalable safety monitoring.

[4] https://drive.google.com/drive/folders/1Ah_3Q2hE8MIuD93e6HLDXtjwC_hqN0H_?usp=sharing

References

[1] Ai-powered fire detection — noematech. https://noema.tech/fire-smoke-detection/, 2024. Accessed: 2025-04-19. 2

[2] Anthropic. Claude 3.7 sonnet. https://www.anthropic.com/claude/sonnet, 2024. Accessed: 2025-04-19. 4

[3] Kriston Burnstein. Firefinder ai: Real-time fire and smoke detection. https://github.com/kriston-burnstein/firefinder-ai, 2024. Accessed: 2025-04-19. 2

[4] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In *European conference on computer vision*, pages 213–229. Springer, 2020. 2

[5] OpenMMLab Contributors. Openmmlab: An open source visual recognition toolbox and benchmark. https://openmmlab.com, 2024. Accessed: 2025-04-21. 3

[6] Pedro Vinícius Almeida Borges De Venâncio, Adriano Chaves Lisboa, and Adriano Vilela Barbosa. D-fire: An image dataset for fire and smoke detection, 2019. Available at https://github.com/gaiasd/DFireDataset. 2

[7] Pedro Vinícius A. B. de Venâncio, Adriano C. Lisboa, and Adriano V. Barbosa. An automatic fire detection system based on deep convolutional neural networks for low-power, resource-constrained devices. *Neural Computing and Applications*, 34(18):15349–15368, 2022. 2

[8] Fares Elmenshawy. Human dataset. https://www.kaggle.com/datasets/fareselmenshawii/human-dataset, 2021. Accessed: 2025-04-22. 3

[9] Armando Fernandes, Andrei Utkin, and Paulo Chaves. Enhanced automatic wildfire detection system using big data and efficientnets. *Fire*, 7(8), 2024. 2

[10] Google Firebase. Firebase cloud messaging. https://firebase.google.com/products/cloud-messaging, 2025. Accessed: 2025-04-19. 4

[11] flutter_mjpeg Developers. flutter_mjpeg: Flutter plugin for mjpeg streaming. https://pub.dev/packages/flutter_mjpeg, 2025. Accessed: 2025-04-19. 4

[12] Abdul-Razak Alhassan Gamani, Ibrahim Arhin, and Adrena Kyeremateng Asamoah. Performance evaluation of yolov8 model configurations, for instance segmentation of strawberry fruit development stages in an open field environment, 2024. 3

[13] GetStream.io. Low latency – what is it and how does it work?, 2025. Accessed: 2025-04-19. 2

[14] Google. Flutter - build apps for any screen. https://flutter.dev, 2025. Accessed: 2025-04-19. 4

[15] Abubakar Hassan and AbdulKadir Audu. Traditional sensor-based and computer vision-based fire detection systems: A review. pages 469–492, 2022. 2

[16] Debapriya Hazra and Yung-Cheol Byun. Upsampling real-time, low-resolution cctv videos using generative adversarial networks. *Electronics*, 9(8), 2020. 2

[17] Anshul Jadon, Mohammad Omama, Yash Varshney, et al. Firenet: A lightweight fire detection dataset, 2019. arXiv preprint arXiv:1903.02772. 2

[18] Chengtuo Jin, Tao Wang, Naji Alhusaini, Shenghui Zhao, Huilin Liu, Kun Xu, and Jin Zhang. Video fire detection methods based on deep learning: Datasets, methods, and future directions. *Fire*, 6(8), 2023. 2

[19] Rahima Khanam and Muhammad Hussain. Yolov11: An overview of the key architectural enhancements, 2024. 3

[20] Rahima Khanam and Muhammad Hussain. What is yolov5: A deep look into the internal features of the popular object detector, 2024. 2

[21] Jichao Li, Shengyu Guo, Liulin Kong, Siqi Tan, and Yuan Yican. An improved yolov3-tiny method for fire detection in the construction industry. *E3S Web of Conferences*, 253: 03069, 2021. 2

[22] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. *European conference on computer vision*, pages 740–755, 2014. 2

[23] Jian Liu, Rui Chen, Xin Hu, Wei Zhang, Yifan Zhu, Lei Zhang, Mingyang Qi, and Liqiang Wang. Multi-task spatio-temporal representation learning for person-centric video understanding. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11855–11865, 2023. 2

[24] Luxonis. Opencv ai kit (oak) with depthai. https://docs.luxonis.com/en/latest/, 2021. Available at https://docs.luxonis.com/en/latest/. 2

[25] Chengqi Lyu, Wenwei Zhang, Haian Huang, Yue Zhou, Yudong Wang, Yanyi Liu, Shilong Zhang, and Kai Chen. Rtmdet: An empirical study of designing real-time object detectors, 2022. 3

[26] Fatma M. Talaat and Hanaa Zaineldin. An improved fire detection approach based on yolo-v8 for smart cities. *Neural Computing and Applications*, 2023. 1

[27] Abdennabi Morchid, Rachid Jebabra, Hassan Qjidaa, Rachid El Alami, and Mohammed Ouazzani Jamil. Agri-tech innovations for sustainability: A fire detection system based on mqtt broker and iot to improve environmental risk management. *Results in Engineering*, 24:103683, 2024. 1

[28] Mohd Halim Mohd Noor and Ayokunle Olalekan Ige. A survey on state-of-the-art deep learning applications and challenges, 2025. 1

[29] OpenAI. Chatgpt-4o. https://openai.com/index/hello-gpt-4o/, 2024. Accessed: 2025-04-19. 4

[30] OpenMMLab. Rtmdet tiny config - rtmdet_tiny_8xb32-300e_coco.py. https://github.com/open-mmlab/mmdetection/blob/main/configs/rtmdet/rtmdet_tiny_8xb32-300e_coco.py, 2023. Accessed: 2025-04-20. 3

[31] Pallets. Flask (version 3.1.x) [computer software]. https://flask.palletsprojects.com/en/stable/, 2025. Accessed: 2025-04-19. 4

[32] Pooja Pandey, Gabriela Huidobro, Luis Filipe Lopes, Anne Ganteaume, Davide Ascoli, Conceição Colaco, Gavriil Xanthopoulos, Theodore M. Giannaros, Rob Gazzard, Georgios

Boustras, Toddi Steelman, Valerie Charlton, Euan Ferguson, Judith Kirschner, Kerryn Little, Cathelijne Stoof, William Nikolakis, Carmen Rodriguez Fernández-Blanco, Claudio Ribotta, Hugo Lambrechts, Mariña Fernandez, and Simona Dossi. A global outlook on increasing wildfire risk: Current policy situation and future pathways. *Trees, Forests and People*, 14:100431, 2023. 1

[33] Rajandran R. Benchmarking the nvidia geforce rtx 4060 for machine learning workloads, 2025. Accessed: 2025-04-19. 3

[34] Joseph Redmon and Ali Farhadi. Yolov3: An incremental improvement, 2018. 2

[35] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection, 2016. 1

[36] A. Saied. Fire dataset. https://www.kaggle.com/datasets/phylake1337/fire-dataset, 2018. Accessed: 2025-04-19. 2

[37] Shuai Shao, Zeming Zhao, Bo Li, Tete Xiao, Gang Yu, Xiangyu Zhang, and Jian Sun. Crowdhuman: A benchmark for detecting human in a crowd. *arXiv preprint arXiv:1805.00123*, 2018. 2

[38] Juan Terven, Diana-Margarita Córdova-Esparza, and Julio-Alejandro Romero-González. A comprehensive review of yolo architectures in computer vision: From yolov1 to yolov8 and yolo-nas. *Machine Learning and Knowledge Extraction*, 5(4):1680–1716, 2023. 2

[39] TP-Link. Tapo c212 pan/tilt home security wi-fi camera. https://www.tp-link.com/in/home-networking/cloud-camera/tapo-c212/, 2025. Accessed: 2025-04-19. 4

[40] Ultralytics. Yolov8 configuration file. https://github.com/ultralytics/ultralytics/blob/main/ultralytics/cfg/models/v8/yolov8.yaml, 2023. Accessed: 2025-04-19. 3

[41] Ultralytics. Yolov11 configuration file. https://github.com/ultralytics/ultralytics/blob/main/ultralytics/cfg/models/11/yolo11.yaml, 2024. Accessed: 2025-04-20. 3

[42] Jonathan P Wanderer, Warren S Sandberg, and Jesse M Ehrenfeld. Real-time alerts and reminders using information systems. *Anesthesiology Clinics*, 29(3):389–396, 2011. 1

[43] Liang Wang, Yifan Wang, Zhifeng Li, and Ruimin Gao. Deep learning for human action recognition: A survey. *Pattern Recognition Letters*, 132:112–120, 2020. 2

[44] Songxi Yang, Qunying Huang, and Manzhu Yu. Advancements in remote sensing for active fire detection: A review of datasets and methods. *Science of The Total Environment*, 943:173273, 2024. 1

[45] Hikmat Yar, Tanveer Hussain, Zulfiqar Khan, Deepika Koundal, Mi Lee, and Sung Baik. Vision sensor-based real-time fire detection in resource-constrained iot environments. *Computational Intelligence and Neuroscience*, 2021:1–15, 2021. 2

[46] Muhammad Yaseen. What is yolov8: An in-depth exploration of the internal features of the next-generation object detector, 2024. 2, 3

[47] Zehnder Clean Air Solutions. Smoke detector false alarm from dust? make it a problem of the past, 2023. Accessed: 2025-04-19. 2

[48] Dawei Zhang. A yolo-based approach for fire and smoke detection in iot surveillance systems. *International Journal of Advanced Computer Science and Applications*, 15(1), 2024. 2

[49] Qixing Zhang, Gaohua Lin, Yong-ming Zhang, Gao Xu, and Jin-jun Wang. Wildland forest fire smoke detection based on faster r-cnn using synthetic smoke images. *Procedia Engineering*, 211:441–446, 2018. 2

Appendix

A. Class Distribution of D-Fire Dataset

The D-Fire Dataset, used for training and evaluating our YOLOv8n smoke and fire detection model, exhibits a class imbalance across its splits. Figures 3, 4, and 5 visualize the distribution of smoke and fire instances in the training (14,122 images), test, and validation (3,099 images) sets. The training set shows a higher frequency of fire instances compared to smoke, a trend consistent in the test and validation sets, potentially influencing the model's stronger smoke detection performance (mAP@0.5:0.95: 0.497 for smoke vs. 0.356 for fire).

Figure 3. Class distribution in the training set (14,122 images).

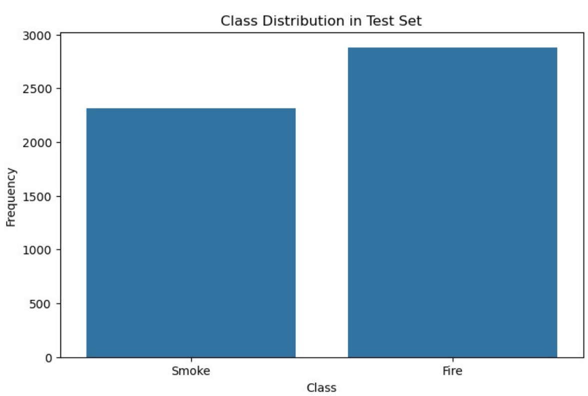

Figure 4. Class distribution in the test set.

B. Sample Detections from D-Fire Dataset

To illustrate the diversity of the D-Fire Dataset and compare the detection performance of YOLOv8n and YOLOv11n, Figures 6 and 7 present grids of sample images with predicted bounding boxes for smoke and fire. The dataset includes a variety of real-world scenarios such as outdoor

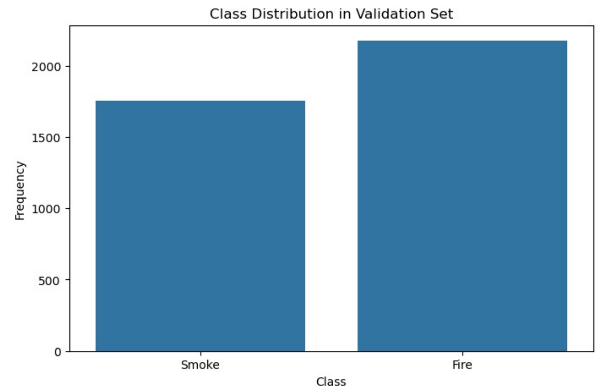

Figure 5. Class distribution in the validation set (3,099 images).

landscapes, urban environments, and varying lighting conditions (e.g., day and night), showcasing each model's robustness.

Figure 6 highlights the performance of the YOLOv8n model. While it successfully detects many smoke and fire instances, certain images (e.g., `WEB03431.jpg`, `WEB03453.jpg`) result in no detections, pointing to its limitations under low-contrast or complex backgrounds.

In contrast, Figure 7 displays predictions from the YOLOv11n model. It exhibits improved localization confidence, detects more instances in difficult settings, and handles challenging cases such as distant or low-visibility smoke more reliably than YOLOv8n.

Figure 6. Sample images from the D-Fire Dataset with YOLOv8n predictions for smoke and fire. Blue boxes indicate smoke, and red boxes indicate fire.

C. Mobile Application Screenshots

The following figures showcase the core user interface screens of the FireSafe mobile application, developed using Flutter. These screens demonstrate the app's capabilities, including authentication, camera feed management, and real-time fire detection alerts.

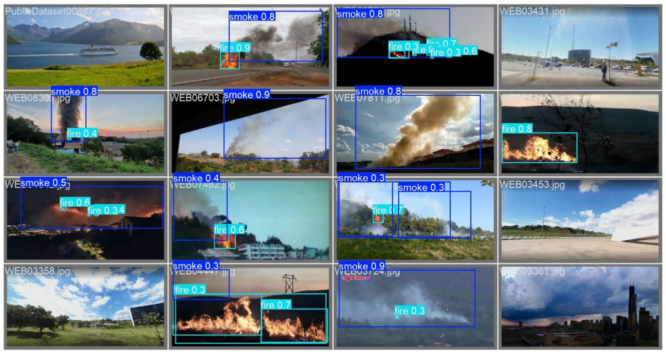

Figure 7. Sample images from the D-Fire Dataset with YOLOv11n predictions for smoke and fire. Blue boxes indicate smoke, and red boxes indicate fire.

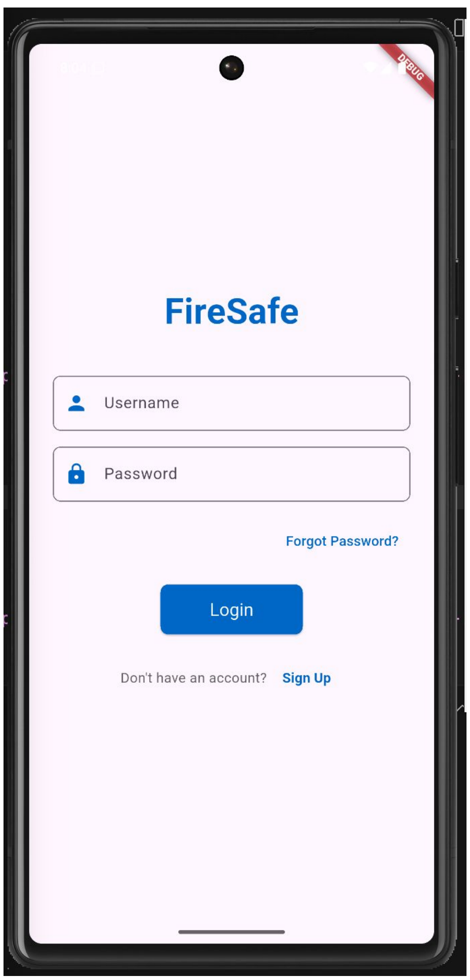

Figure 8. Login Screen — Allows users to securely log into the FireSafe app.

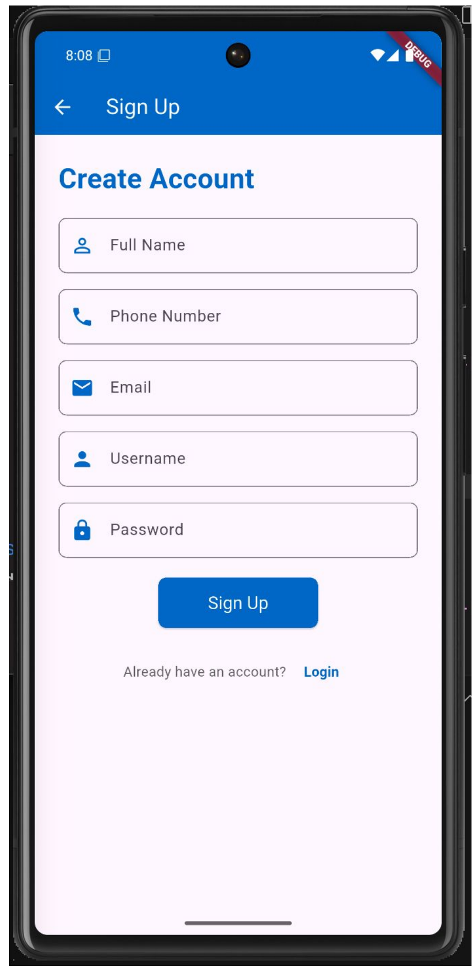

Figure 9. Sign-Up Screen — Enables new users to create an account using personal details.

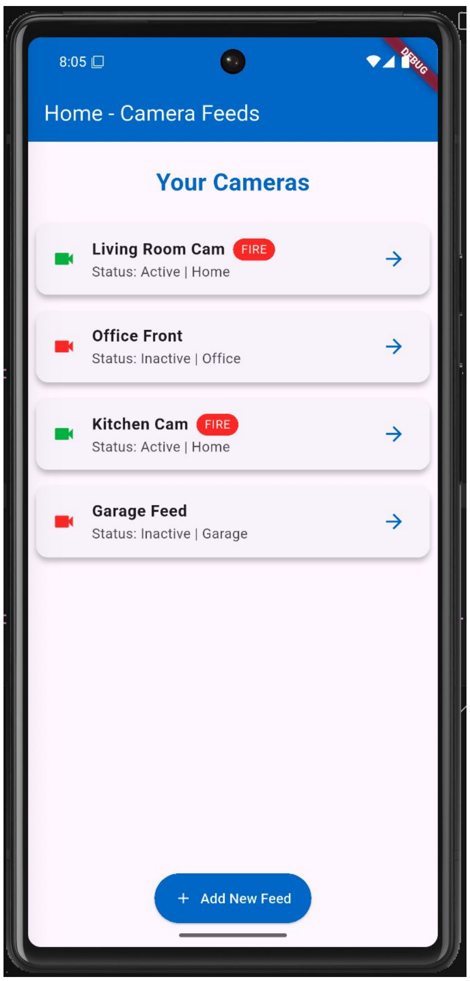

Figure 10. Home Screen — Displays all camera feeds with their statuses and fire detection indicators.

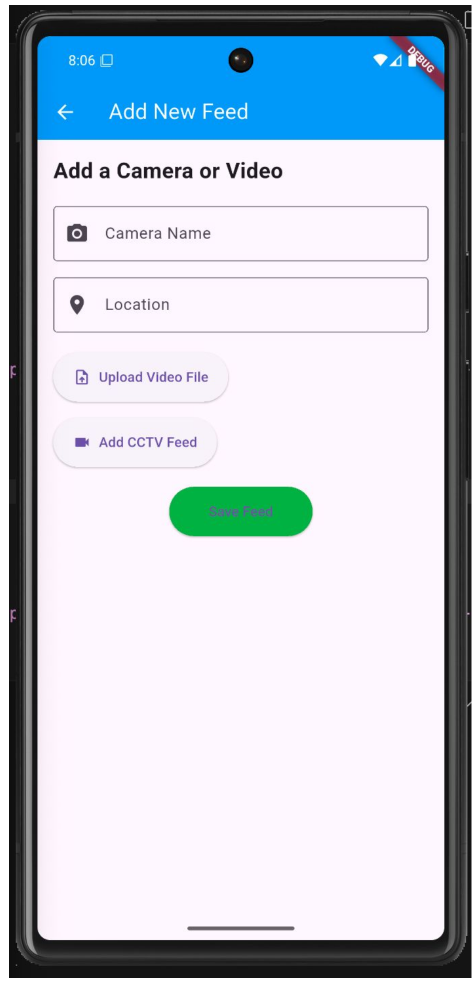

Figure 11. Add Feed Screen — Allows users to add a camera or upload a video for monitoring.

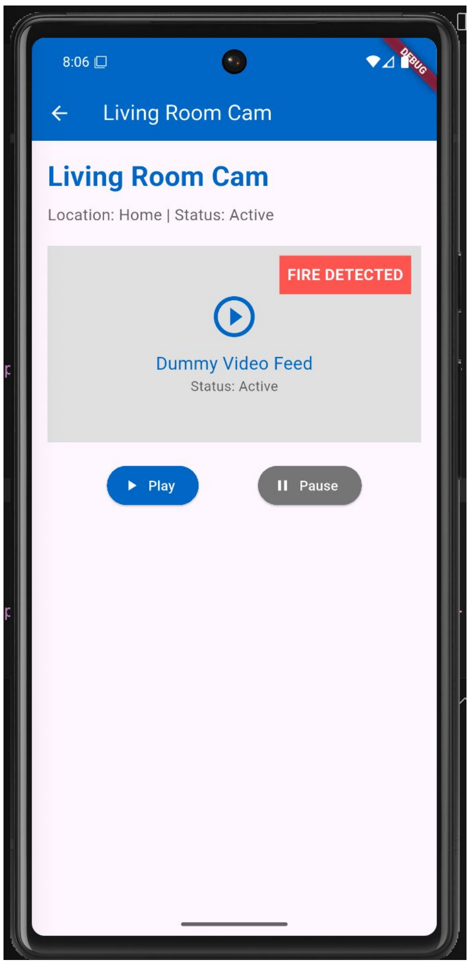

Figure 12. Camera View (Fire Detected) — Annotated video feed showing an active fire alert.

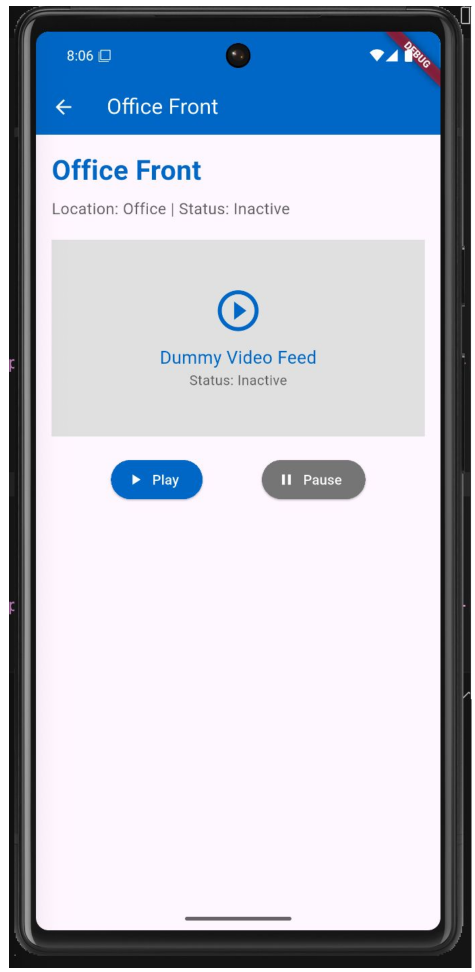

Figure 13. Camera View (Inactive) — Example of an inactive feed with no detected fire.