Bonus Use Case Proposal:

Team: Emperors

Hackathon: BuildWithDelhi 2.0

---

1. Introduction

Space missions are high-stakes environments where every piece of equipment must be accounted for. Traditional manual checks are time-consuming and prone to human error. To address this, our bonus use case evolves the trained YOLOv8 detection model into two deployable solutions: **Safety Detection Web App** for centralized management and a **Real-Time Live Detection** for continuous surveillance. These applications provide mission control with an automated safety monitoring system capable of detecting, tracking, and alerting in real time. Their combined functionality increases reliability, efficiency, and preparedness.

---

2. Application Description

Applications Implemented:

1. Safety Detection Web App

A modern web application integrating detection results with user-friendly features for remote monitoring.

- **Backend:** FastAPI (safety-detection-app/backend/main.py) processes detection requests and serves results.
- **Frontend:** React (safety-detection-app/frontend/src/) provides an interactive interface.
- **Core Features:**
  - Upload images or videos and receive detection outputs instantly
  - Visualization of detection results with bounding boxes and confidence scores
  - Detection history stored with timestamps for analysis
  - User feedback mechanism to flag incorrect detections
- **Usage:** Launch FastAPI backend and React frontend, then open http://localhost:5173.

These applications collectively support both on-site real-time monitoring and remote data management.

2. Real-Time Live Detection

A standalone Python-based client enabling live monitoring through webcams, IP cameras, or pre-recorded video streams.

- **Script:** Real-time-live-detection/realtime_detection.py
- **Function:** Captures frames, performs YOLOv8 inference, and overlays detection results on live video.
- **Advanced Features:**
  - GPU/CPU auto-selection with performance tuning
  - Dynamic confidence and IoU threshold adjustments for precision control
  - Optional saving of annotated video streams
  - Color-coded bounding boxes based on confidence levels
  - FPS and device information overlay for diagnostics
  - Compatibility with DroidCam and other IP camera sources
- **Usage Examples:**

```
python realtime_detection.py --source 0  # Webcam
python realtime_detection.py --source http://192.168.1.2:4747/video  # IP Camera
python realtime_detection.py --output output.mp4  # Save annotated video
python realtime_detection.py --device cpu  # Force CPU inference
python realtime_detection.py --conf 0.6  # Set confidence threshold
```

- **CLI Options:** --source, --output, --device, --conf, --iou, --max-det

---

3. Technical Implementation

The system combines state-of-the-art AI detection with efficient software architecture:

- **YOLOv8 (Ultralytics):** High-accuracy, real-time detection backbone.
- **OpenCV:** Handles video capture, processing, and frame annotation.
- **Threading:** Improves frame processing throughput for live streams.
- **FastAPI:** Lightweight, asynchronous backend serving detection and user management requests.
- **React Frontend:** Dynamic and responsive interface for a seamless user experience.

- **REST Endpoints:** Enable communication between the web app frontend and backend detection engine.

The design ensures scalability, easy maintenance, and smooth performance on a range of hardware.

---

4. Model Update Strategy with Falcon

Our solution uses Falcon’s synthetic data pipeline to maintain peak accuracy:

- **Synthetic Data Updates:** Falcon generates new datasets reflecting equipment variations and environmental changes.
- **Incremental Training:** The YOLOv8 model is fine-tuned with new data while retaining existing knowledge.
- **Deployment Workflow:** Updated weights are packaged and seamlessly integrated into both applications, ensuring continuous improvement without downtime.

This strategy guarantees that the monitoring system stays adaptive to evolving mission requirements.

---

5. Demo Evidence

- **Web App Demo:** Watch here – Demonstrates UI interactions and detection on uploaded media.
- **Real-Time Detection Demo:** Watch here – Shows the live detection client processing station video streams.
- **Screenshots:** Include samples of detection overlays, React UI components, and alert logs.

These demos confirm the successful deployment and usability of both applications.

---

6. Future Enhancements

Planned upgrades to strengthen and extend capabilities:

- **Edge Deployment:** Optimize model for devices like NVIDIA Jetson to run onboard.
- **Multi-Stream Monitoring:** Handle multiple video sources simultaneously.
- **WebSocket Integration:** Provide live streaming of detection results to the web interface.

- **Automated Falcon Retraining:** Fully automate the retraining and redeployment pipeline.
- **Extended Object Classes:** Add detection for additional critical tools and supplies.

These improvements will enhance scalability, performance, and operational coverage.

---

7. Conclusion

The Real-Time Live Detection client and Safety Detection Web App exemplify the practical application of our trained YOLOv8 model. By coupling advanced AI with intuitive user interfaces, these tools deliver a robust solution for automated safety monitoring. The Falcon-powered update strategy ensures that the system remains adaptive, accurate, and valuable for future space missions.

Deliverables:

- Real-Time Detection Script: Real-time-live-detection/
- Web App: safety-detection-app/
- Demo Videos: Linked above
- Falcon Integration Plan: Included

Prepared by Team Emperors for the Bonus Use Case Submission