Pizza Store Scooper Violation Detection System

Overview

This system is designed to automatically detect hygiene violations in pizza restaurants by identifying instances where an employee uses their hands directly in food containers instead of using designated scoopers. The solution uses a lightweight YOLOv8-nano object detection model for efficiency and accuracy and is built using a microservices architecture for scalability and modularity.

Table of Contents

1. System Architecture

The system is divided into the following microservices:

- AI Module: Handles training and inference of the YOLO model.
- Flask Backend: Manages video uploads, initiates detection, and returns predictions.
- Angular Frontend: Provides a user interface for uploading videos and visualizing results.

2. Technologies Used

Component	Technology
Detection	YOLOv8-nano
Inference	Python, OpenCV
Model Hosting	Flask
Frontend	Angular
Dataset	Roboflow / Custom
Video I/O	OpenCV, gdown

3. AI Module

3.1 Model Choice

Initially, a YOLOv12 checkpoint was considered for the detection task. However, it produced suboptimal results and required high computational resources. To balance performance and efficiency, the system was shifted to **YOLOv8-nano**, which offered the following benefits:

- Faster training and inference times.
- Significantly reduced computational and memory requirements.
- Retains the robust performance of the YOLO architecture.

This model was more than sufficient for the task at hand, which primarily focused on detecting people, scoopers, hands and pizza within the food-handling area.

3.2 Training Details

- **Model Used:** yolov8n.pt
- **Data Source:** Custom-labeled dataset prepared via Roboflow. Classes included: hands, scooper, pizza and person.
- **Training Parameters:**
    - Epochs: 100
    - Image Size: 640
    - Batch Size: 16
    - Device: GPU (device=0)

3.3 Inference Logic

The output consists of detected bounding boxes and class labels for each frame of the video.

4. Violation Detection Logic

The violation detection logic is applied post-inference and works as follows:

1. **Handcrafted Regions of Interest (ROIs)** were defined for food containers inside the video frame.
2. The system tracks the movement of hands and scoopers within each frame.
3. If a **person's hand enters a container's ROI** and **no scooper is present**, the system flags it as a **violation**.

5. Web Application

The system is deployed as a web application using a **microservices-based architecture**, consisting of a Flask backend and an Angular frontend.

5.1 Frontend (Angular)

The frontend is implemented using **Angular**, which is a powerful framework for building scalable, modular user interfaces. It was chosen because:

- Angular facilitates separation of concerns in a microservices architecture.
- Components and modules are easily maintainable and testable.
- It supports rapid development and smooth integration with APIs.

The frontend allows users to upload videos, view progress, and receive violation detection results visually.

5.2 Backend (Flask)

The backend, built with **Flask**, serves the following functions:

- Receives video uploads via an API endpoint.
- Calls the YOLO inference service.
- Applies the ROI violation detection logic.
- Sends responses with results and optionally processed video frames for visualization.

Flask was chosen due to its simplicity, flexibility, and ease of integration with both the AI model and the Angular frontend.