Computer Vision Task — Pizza Store Scooper Violation Detection (Microservices-Based Architecture)

📌 Overview

This project is focused on building a **Computer Vision system** for a **pizza store** to monitor hygiene protocol compliance. Specifically, the system detects **whether workers are using a scooper** when picking up certain ingredients (like proteins) from designated areas (ROIs - Regions of Interest). Any action of picking up these ingredients **without a scooper** will be flagged as a **violation**.

🎯 Objective

Design and implement a **microservices-based system** that:

1. **Ingests video** from cameras in the pizza store.
2. Identifies **interactions with ROIs**, such as the protein container not all containers are important only the one he grapes with scooper in the videos.
3. Determines **whether a scooper was used**.
4. Flags any **violations** (ingredient grabbed without scooper from ROIs only).
5. Handle the case when he can get his hand at ROI but not getting anythings (like cleaning)
6. Handle the case when two works on the pizza table at the same time
7. Display the video with **detection results to a frontend** in real-time.

📦 Resources Provided

- **Raw data**

  unannotated videos link

- **Dataset based on the videos :**
  Pizza Store Dataset on Roboflow
  Use this dataset to train the model again if you want. All videos annotated except these

: (Video.mkv , Video 1.mkv , Video 2.mkv)

- **Pretrained Model:**
  Download the pretrained model
  The model is trained to detect objects relevant to the task (Hand,Person,Pizza,Scooper). You can use it directly in your inference microservice or finetune it again.

  The model is YOLO 12 medium trained on the dataset above with **1254 images**

- Sample videos to test your solution on :
  - Sah w b3dha ghalt.mp4 has real violations = 1
  - Sah w b3dha ghalt (2).mp4 has real violations = 2
  - Sah w b3dha ghalt (3).mp4 has real violations = 1

🧱 Microservices Architecture Requirements

You are expected to use a **microservices-based approach** to build the system. The architecture should be modular and decoupled for scalability and maintainability.

Feel free to add extra microservices if needed.

🖼️ 1. Frame Reader Service

- Reads video frames from a video file or RTSP camera feed.
- Publishes frames to a **message broker**.
- **Suggested tools:** OpenCV,Gstreamer, deep stream, FFmpeg or any library

📨 2. Message Broker

- Facilitates communication between services.
- **Choose one:** Apache Kafka or RabbitMQ.
- Handles buffering and stream management.

🔍 3. Detection Service

- Subscribes to the message broker.
- Performs object detection using the provided pretrained model.
- Checks the logic:
    - If a hand took an ingredient and put on pizza (e.g., protein cargo) **without a scooper**, log it as a violation.

- save frame path that has violation + (bounding boxes, labels, timestamp) of violations in database.
- Sends results (bounding boxes, labels, violation status) to the streaming service.

🌐 4. Streaming Service

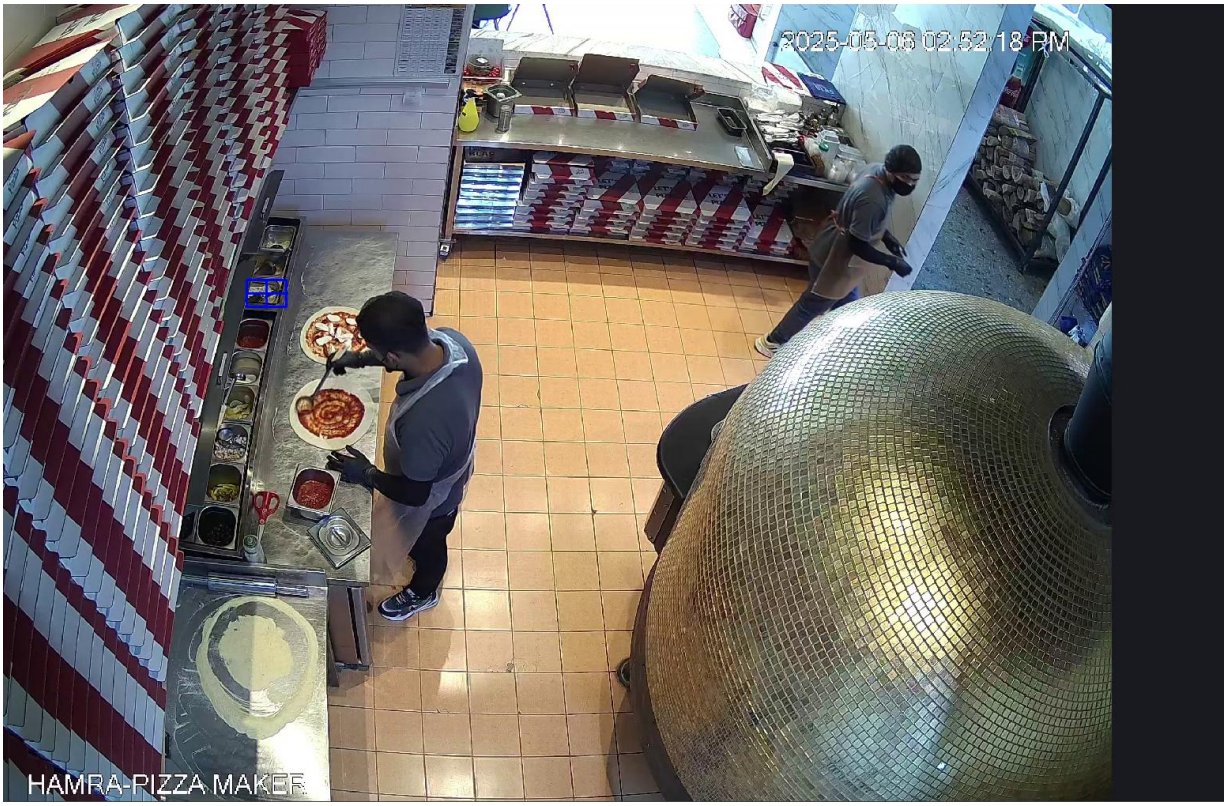
- Serves detection results via:
    - REST API for metadata (number of violations in the video) to display on the frontend
    - WEBRTC or WebSocket or MJPEG stream for real-time video displaying with detections drawn on it .
- **Suggested frameworks: Any backend (**e.g., Flask or FastAPI ,etc).

💻 5. Frontend UI

- Receives detection results and visualizes them.
- Should highlight:
    - Bounding boxes of detected objects
    - ROI areas
    - Violation events (red box or alert)

- **Suggested frameworks: Any web interface using** HTML/CSS/JS , frontend framework (React.js,Vue.js,etc) or easy to deploy framework like (streamlit,gradio) but not recommended for flexibility.

Logic for Violation Detection



1. User defines **ROIs** (Regions of Interest) for each critical zone (e.g., protein cargo).
2. For each frame:
    - If a hand enters ROI and didn't grap with scooper when returned to any pizza → count **as a violation**.
    - If a **scooper is present** and used to pick up the ingredient → **no violation**.
3. Count and display the total number of violations on frontend.

For this video showing in the image the number of violations is : **1**

Link of example video : Sah b3dha ghalt.mp4

☑ Deliverables

- Functional system with microservices architecture as described above.
- Source code with clear documentation.
- README with setup instructions.
- UI showing:
    - Video stream
    - Display detection on the video in Real-time
    - Count of violations
- Recorded video of the system working.
- Docker Compose (optional but preferred) for easy deployment.

🚀 Bonus Points

- Use Docker & Docker Compose for service orchestration.
- When choosing best framework or library for each service.