Real-Time YOLO-Based Object Detection System

with Interactive Gradio Web Interface

Computer Vision Project

BS Artificial Intelligence

Project Team Members

- Malik Anees Ahmed (BSAI-132)
- Mudassir (BSAI-196)
- Niaz (BSAI-197)
- Abdulla (BSAI-306)
- Faizan (BSAI-307)

Submitted to:	Mam Ayesha Safdar
Submission Date:	17 December 2024
Project Video Link:	`https://drive.google.com/file/d/1WBuVHWfEuhNPxMjgkWFFtnmypi3NGfk3/view?usp=sharing`
GitHub Repository Link:	`https://github.com/MalikAnees530/yolo-gradio-object-detection-cv-project`

1 Introduction

Object detection is a fundamental computer vision task that involves identifying and localizing objects within images or video frames. Unlike simple image classification, object detection provides both the class label and spatial coordinates of detected objects, making it essential for numerous real-world applications.

Real-time object detection has become increasingly important in modern society, powering critical applications across various domains. In surveillance systems, it enables automated monitoring of public spaces, detecting suspicious activities or unauthorized access. Traffic monitoring systems utilize object detection to count vehicles, detect violations, and manage traffic flow efficiently. Autonomous vehicles rely heavily on real-time detection to identify pedestrians, other vehicles, traffic signs, and obstacles, making split-second decisions for safe navigation. Additionally, retail analytics, industrial automation, and smart city infrastructure all benefit from accurate and fast object detection capabilities.

The primary objective of this project is to implement a real-time YOLO-based object detection system with an interactive Gradio web interface deployed on Google Colab. This project aims to demonstrate the practical application of state-of-the-art computer vision techniques in an accessible, user-friendly format. By leveraging the speed and accuracy of the YOLO (You Only Look Once) architecture and the simplicity of Gradio for building web interfaces, we create a system that allows users to upload images and receive instant detection results with adjustable parameters.

The proposed system is designed with the following goals:

- Provide a clean and intuitive interface for experimenting with object detection.
- Deliver real-time visual feedback through annotated images and textual summaries.
- Serve as an educational and demonstrative tool for modern computer vision techniques.

2 Literature Review

2.1 Classic Object Detection Methods

The evolution of object detection has progressed through several generations of algorithms. Early methods like the Region-based Convolutional Neural Network (R-CNN) introduced by Girshick et al. in 2014 revolutionized the field by combining region proposals with deep learning. R-CNN generates approximately 2,000 region proposals using selective search, extracts features using a CNN, and then classifies each region. While effective, R-CNN suffered from slow inference times due to processing each region independently.

Fast R-CNN and Faster R-CNN improved upon this architecture by introducing Region Proposal Networks (RPNs) and sharing computation across proposals. The Single Shot Detector (SSD), developed by Liu et al., took a different approach by eliminating the region proposal stage entirely and performing detection in a single forward pass through the network. SSD uses multiple feature maps at different scales to detect objects of various sizes, achieving a better balance between speed and accuracy.

2.2 YOLO Family of Detectors

The YOLO (You Only Look Once) family represents a paradigm shift in object detection. Introduced by Redmon et al. in 2016, YOLO reframes object detection as a single regression problem, directly predicting bounding boxes and class probabilities from full images in one evaluation. This approach enables real-time processing at impressive speeds while maintaining competitive accuracy.

The YOLO architecture has evolved through multiple versions, each introducing significant improvements:

- **YOLOv2 (YOLO9000):** Added batch normalization, anchor boxes, and multi-scale training.
- **YOLOv3:** Introduced feature pyramid networks for improved multi-scale detection.
- **YOLOv4 and YOLOv5:** Incorporated CSPDarknet backbones, PANet, and advanced data augmentation techniques.

These developments have continually improved both accuracy and speed, making YOLO-based detectors popular choices for real-time applications.

2.3 YOLOv8 and Model Choice

YOLOv8, released by Ultralytics in 2023, represents the latest advancement in the YOLO series. It introduces an anchor-free detection head, improved feature extraction with a new CSPDarknet backbone, and enhanced training strategies. YOLOv8 achieves state-of-the-art performance across multiple object detection benchmarks while maintaining real-time inference speeds.

For this project, the YOLOv8n (nano) variant is selected due to the following advantages:

- Excellent trade-off between speed and accuracy, suitable for real-time use in Google Colab.
- Clean and well-documented Python API provided by the Ultralytics library.
- Pre-training on the COCO dataset with 80 object classes, enabling broad applicability without extra training.
- Compact model size, resulting in fast loading and inference, which is ideal for interactive web applications.

2.4 Gradio for Rapid ML UI Deployment

Gradio is an open-source Python library that enables rapid creation of web-based user interfaces for machine learning models. It offers:

- Pre-built components for common input and output types such as images, text, and audio.
- Automatic generation of shareable web interfaces and REST APIs.
- Simple deployment options, including public sharing links and integration with cloud platforms.

In this project, Gradio is used to expose the YOLO model through an intuitive interface where users can upload images, adjust detection thresholds, and immediately visualize the results.

3 Solution Approach

3.1 Overall Architecture

The proposed object detection system follows a streamlined pipeline designed for efficiency and user-friendliness:

1. **Input stage:** Users upload an image through the Gradio interface. The image is accepted in NumPy array format and converted to a PIL image for processing.
2. **Model inference:** The YOLOv8 model processes the input image, detecting objects and generating bounding box predictions with confidence scores and class labels.
3. **Post-processing:** Detection results undergo non-maximum suppression to eliminate redundant overlapping boxes. The system then generates an annotated image with bounding boxes and labels, and computes object-count statistics.
4. **Output display:** The Gradio interface presents both the annotated image and a text summary of detected objects to the user.

3.2 Design Choices

Key design decisions for the system include:

- **Pre-trained YOLOv8n on COCO:** Utilizes a general-purpose model trained on 80 everyday object categories, removing the need for custom training.
- **Interactive thresholds:** Confidence and IoU sliders allow users to control the sensitivity and strictness of detections without modifying code.
- **Textual summary:** A concise text summary lists each detected object class along with its count, supporting quick quantitative analysis.

4 Implementation

4.1 Development Environment

The project is implemented in Google Colab, a cloud-based Jupyter notebook environment that provides free access to GPU resources. Python 3 is used with the following main libraries:

- **ultralytics:** Official YOLOv8 implementation for model loading, inference, and visualization.
- **gradio:** Framework for building the interactive web interface.
- **numpy:** Numerical operations and array manipulations.
- **Pillow (PIL):** Image loading, processing, and conversion.
- **torch:** PyTorch deep learning framework (dependency of `ultralytics`).

4.2 Screenshots of Implementation

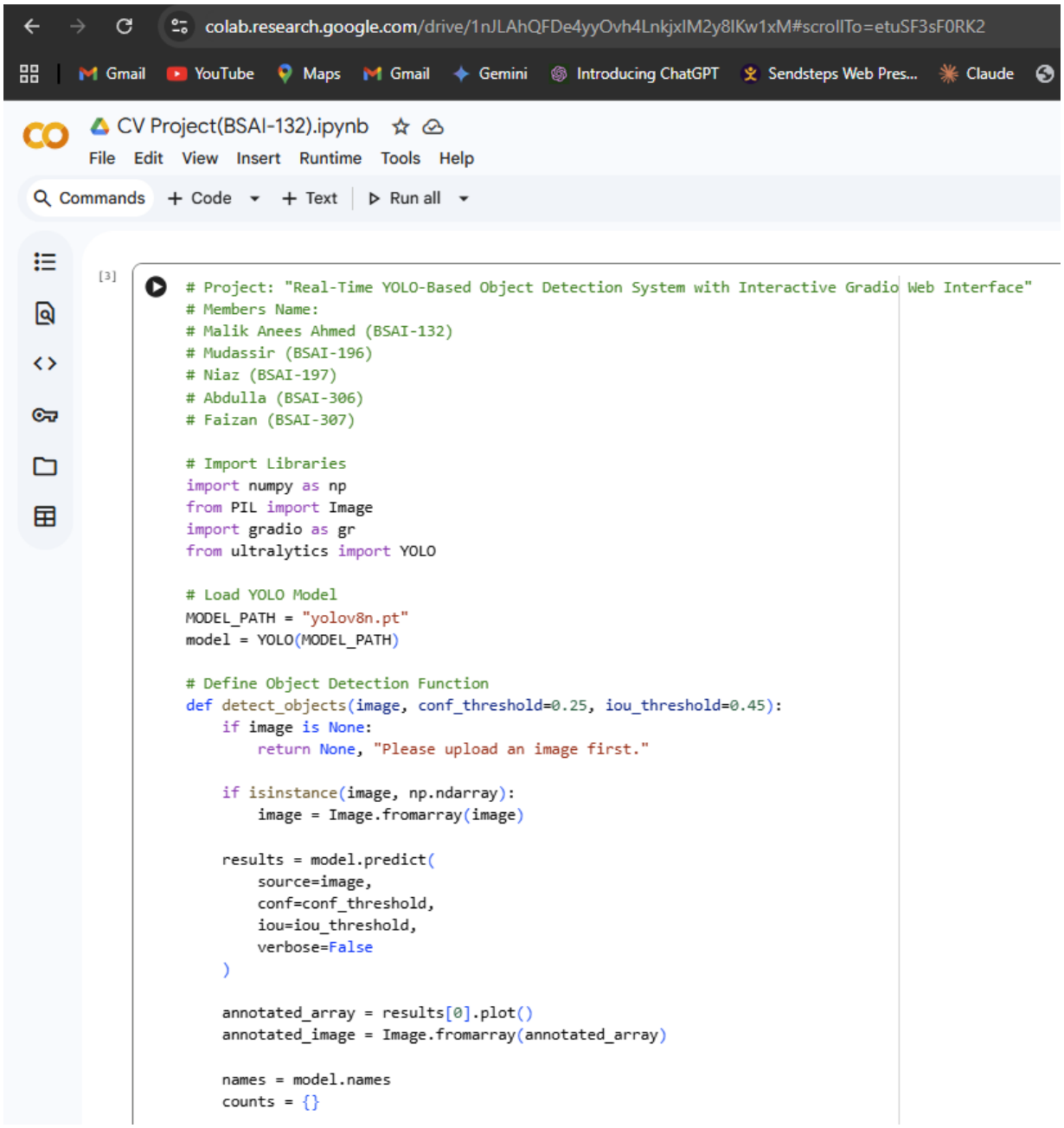

Figure 1: Google Colab notebook showing the YOLO model loading and `detect_objects` implementation.

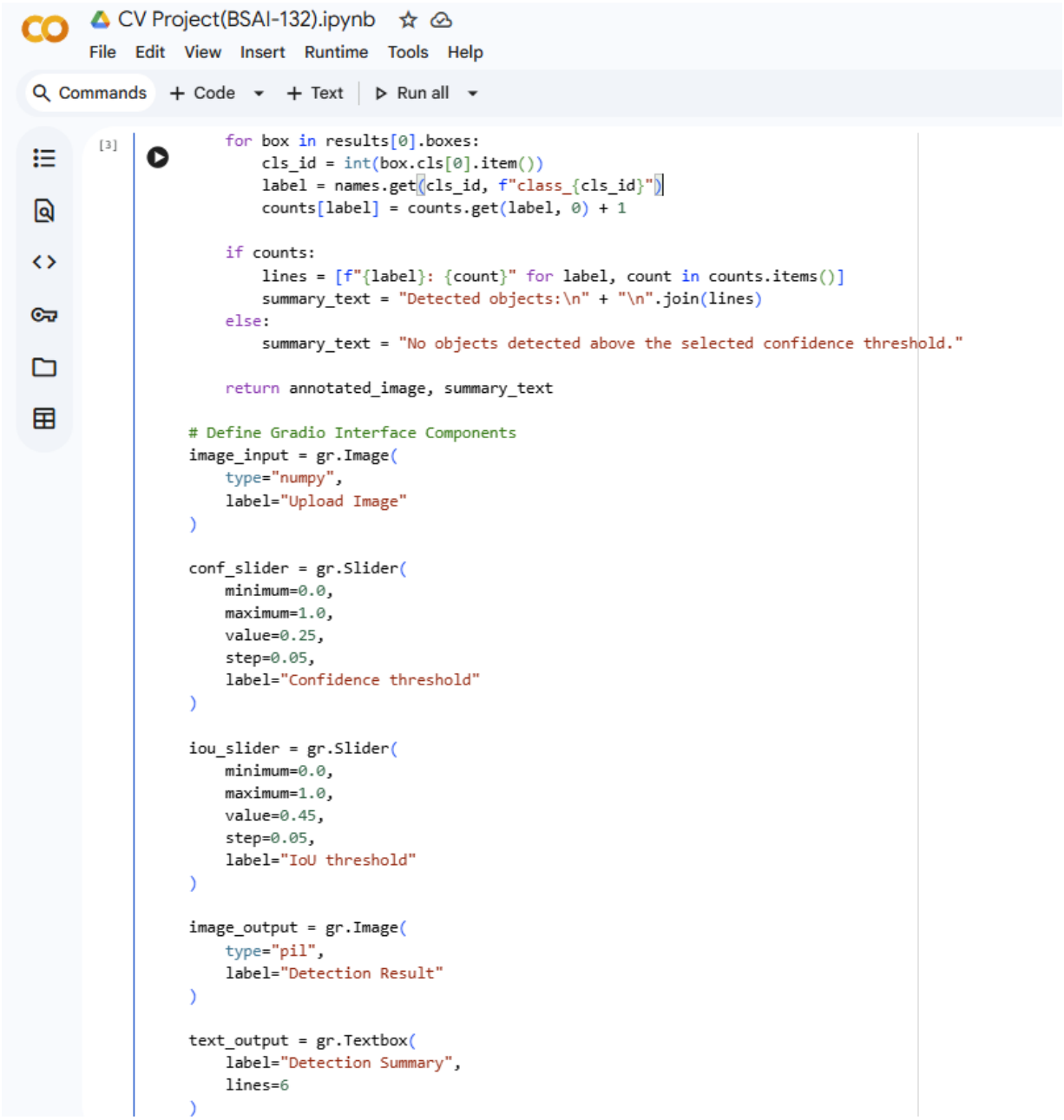

Figure 2: Configuration of the Gradio interface for uploading images, setting thresholds, and displaying detection results.

5 Results and Discussion

5.1 Experimental Results

The object detection system was evaluated on a set of images representing different scenarios, including busy street scenes, indoor environments, and park or crowd scenes. Table 1 summarizes representative qualitative results.

Image Type	Conf.	IoU	Main Detections	Observations
Street scene	0.25	0.45	Cars, persons, traffic lights	Accurate detection of vehicles and pedestrians; very small distant objects occasionally missed.
Indoor room	0.30	0.45	Couch, chairs, TV, plants	Correct localization of main furniture; minor false positives on cluttered areas.
Crowd / park	0.25	0.45	Persons, animals	Strong detection of people and animals, with some misses under heavy occlusion.

Table 1: Summary of representative detection results.

5.2 Visualization of Results

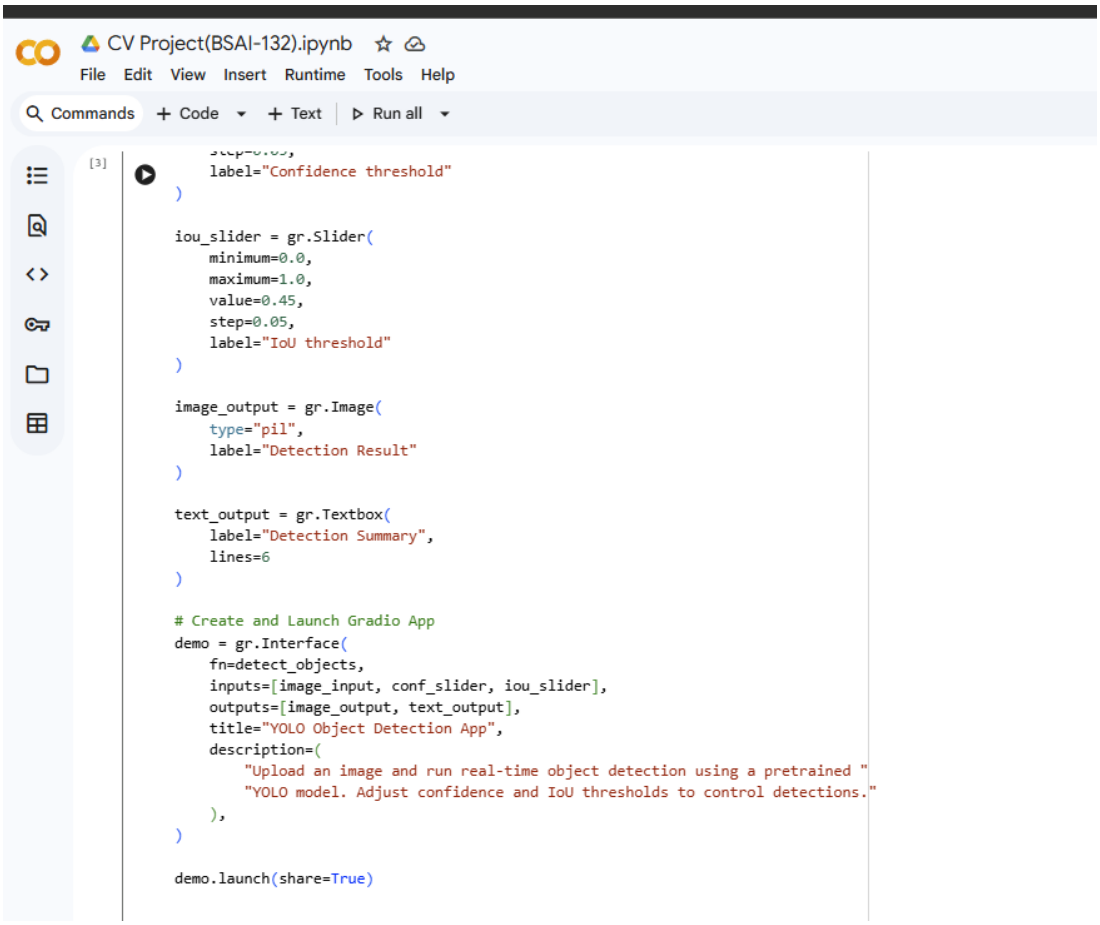

Figure 3: Detection result for a busy street scene using the YOLOv8n model.

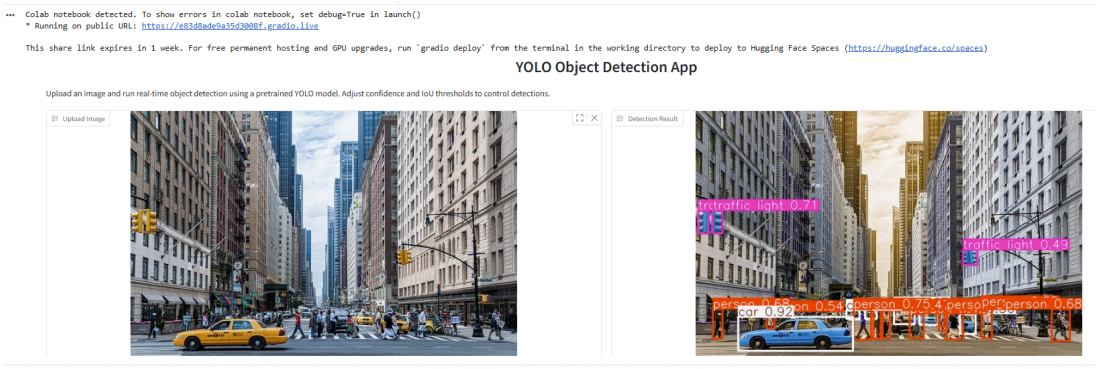

Figure 4: Detection result for an indoor environment, highlighting furniture and other objects.

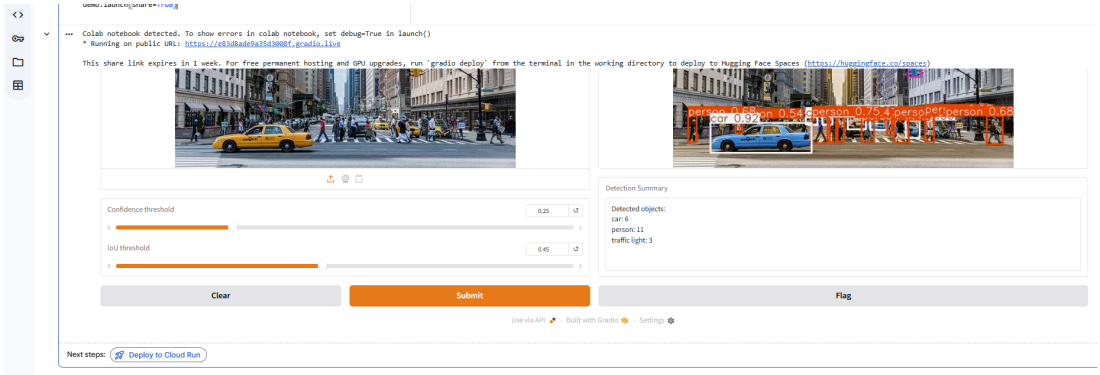

Figure 5: Detection result for a park and crowd scene, showing multiple persons and objects.

5.3 Performance Observations

Inference speed on Google Colab's GPU averaged tens of milliseconds per image for typical resolutions, enabling real-time interaction through the Gradio interface. CPU inference was slower but still acceptable for offline analysis. The model demonstrated robust performance across various lighting conditions and backgrounds. However, very small or heavily occluded objects remained challenging, and the detection quality was sensitive to the chosen confidence and IoU thresholds.

6 Conclusion

6.1 Project Summary

This project demonstrates the implementation of a real-time object detection system using the YOLOv8 architecture integrated with an accessible Gradio web interface. The system meets its primary objective of providing a user-friendly platform that makes advanced computer vision capabilities easy to explore. It effectively detects and localizes objects across diverse scenarios and provides both visual and textual outputs for interpretation.

6.2 Limitations

- The model is limited to the 80 object classes present in the COCO dataset and cannot recognize objects outside this set.

- Very small objects, especially those occupying only a few pixels in the image, are frequently missed.
- Heavily occluded objects are rarely detected successfully, which reduces performance in crowded scenes.

6.3 Future Work

Future extensions of this project may include:

- Training YOLOv8 on a domain-specific custom dataset, such as local traffic signs or classroom environments, to improve performance on specialized tasks.
- Adding support for real-time video and webcam streams using Gradio's video or WebRTC components, enabling continuous detection over time.
- Deploying the application to platforms such as HuggingFace Spaces or cloud infrastructure to provide persistent public access and turn this course project into a reusable demonstration tool.