# Report: Real-time Object Detection Pipeline

Rushikesh Waghmare

### GitHub Repository

https://github.com/iamrealrushi/Real-time-Object-Detection-Pipeline-

## 1 Implementation Approach

- **Framework & Model:** Utilized the `YOLOv8s` model from the Ultralytics library for real-time object detection.
- **Data Source:** Real-time webcam video feed captured using OpenCV (`VideoCapture(0)`).
- **Processing Pipeline:**
  - Each frame passed to the YOLOv8 model for detection.
  - Bounding boxes and labels manually drawn for each detected object.
  - Total objects per frame counted and displayed.
  - Frames displayed in a window and saved to `output.avi`.
- **Optimization:** Used image size of 640 and confidence threshold of 0.25 to balance detection quality and real-time speed.

## 2 Results

- **Demo Video:** Detection output saved as `output.avi`.
- **Detected Classes:** Supports 10+ common COCO object classes via YOLOv8.
- **Performance Metrics:**
  - **Frame Rate:** $\sim$16 FPS (meets requirement of $>$15 FPS).
  - **Detection Confidence:** Displayed live for each object.
  - **Object Count:** Real-time count shown in each frame.

## 3 Challenges & Solutions

- **Challenge:** Maintaining real-time performance without degrading accuracy.
- **Solution:** Used the lightweight `yolov8s.pt` model and fine-tuned confidence and frame resolution settings.

## 4 Future Improvements

- Add GPU acceleration support for higher frame rate.
- Incorporate object tracking IDs for trajectory visualization.
- Use asynchronous processing for lower latency.