1. Introduction

This project involves implementing a real-time object detection system using **YOLOv8**, a state-of-the-art deep learning model, to detect and annotate objects in a video stream. The annotated output is saved as a new video for analysis and visualization. The system is built using **Python**, leveraging **OpenCV** for video processing and **Ultralytics' YOLOv8 API** for model inference.

---

2. Objective

- Detect multiple objects (e.g., humans, vehicles, animals) in each frame of a video.
- Visualize the detections with bounding boxes, labels, and confidence scores.
- Save the annotated frames into a new output video file.

---

3. Tools & Technologies

Tool / Library	Purpose
Python	Programming language
OpenCV	Video processing, annotation, and export
Ultralytics YOLOv8	Object detection model
Matplotlib	(Optional) Visualization of results
Kaggle	Environment for execution and video input/output

---

4. Methodology

4.1 Model Initialization

The `YOLOv8n.pt` (nano version) model is loaded for efficient real-time detection with a good balance of speed and accuracy.

```
python
CopyEdit
model = YOLO("yolov8n.pt")
```

4.2 Video Input and Output Setup

- The video is read frame-by-frame using `cv2.VideoCapture`.
- A `cv2.VideoWriter` object is created to save the processed frames with detections.

4.3 Frame Processing Loop

For each frame:

1. The model performs object detection.
2. Predictions are filtered using a confidence threshold of 0.5.
3. Bounding boxes, labels, and confidence scores are drawn.
4. The modified frame is written to the output video.

4.4 Finalization

Once all frames are processed, the video capture and writer objects are released.

---

5. Results

- The system successfully detects and annotates multiple object classes.
- The output is saved in `detected_output.avi`.
- The model achieves **real-time performance** using the nano variant.

---

6. Applications

- Surveillance systems
- Traffic and pedestrian monitoring
- Industrial safety systems
- Wildlife observation
- Preprocessing for video analytics tasks

---

7. Challenges

- Balancing detection accuracy with real-time speed.
- Ensuring video frame resolution compatibility with the model and output writer.
- Handling cases with low lighting or occluded objects.

---

8. Skills Demonstrated

- **Computer Vision & Deep Learning**
- **YOLOv8 integration**
- **Python scripting**
- **OpenCV-based video handling**
- **Model inference optimization**
- **End-to-end system development and testing**

---

9. Your Role

As the **sole developer**, you were responsible for the complete project pipeline:

- Designing the solution
- Selecting and integrating YOLOv8
- Implementing the real-time detection logic
- Handling video input/output
- Testing and debugging the system
- Generating output for practical use

---

10. Conclusion

This project successfully demonstrates a complete real-time object detection system using YOLOv8. It highlights the practical use of deep learning in computer vision and provides a foundation for further work in activity recognition, tracking, or automated video surveillance.