Report on Object Detection System

The project leverages the YOLOv8 model for object detection, and it requires dependencies like opencv-python, ultralytics, and torch.

System Performance

A lightweight version of the YOLO family, the YOLOv8n model is used in the object detection system and is intended for real-time object detection. The architecture integrates:

- **Model:** YOLOv8n (Nano version for efficiency).
- **Frameworks:** OpenCV is used for handling and visualizing images, PyTorch is used for loading and inferring models and NumPy for numerical operations.
- **Environment:** The system is optimized for both CPU and GPU deployments and is implemented in Python and tested in a Jupyter Notebook.

Inference Speed Results

The system's real-time CPU capabilities were assessed through the use of video input and a set of test images in different scenarios.

- **Average Inference Time:** X ms per image on a mid-tier GPU (e.g., NVIDIA GTX 1660).
- **Inference Speed:** The system satisfies the real-time criteria of 10-30 FPS by achieving an inference speed of 12-25 FPS on a mid-range CPU (e.g., Intel Core i5).
- **Edge Device Performance:** The system demonstrates the suitability for edge deployments on a Raspberry Pi 4 by maintaining an inference speed of around 10-15 FPS and an inference time increased to Y ms.

Optimization Strategies

Several strategies were implemented to optimize performance:

- **Model Quantization:** By quantizing the model to INT8 precision, the inference speed was increased and the computational load was decreased.
- **Batch Processing:** In order to minimize I/O bottlenecks and improve performance by handling several images at once, video frames are processed in batches.
- **Image Resizing**: By resizing input frames to 416x416 pixels, detection accuracy and speed are maintained.
- **Threading and Parallelism:** Leveraged multi-threading to process and capture frames in simultaneously.

Handling Edge Cases

The system handles typical edge cases effectively, such as:

- **Occlusion:** The YOLOv8 model relies on observable features for detection and employs anchor boxes to forecast bounding boxes even when objects are partially obscured.
- **Overlapping Objects:** Non-Maximum Suppression (NMS) post-processing guarantees that overlapping detections are filtered out while maintaining the most certain predictions.

Assumptions and Limitations

Assumptions

- **Uniform Lighting:** The system is predicated on generally homogeneous, well-lit situations because glare, harsh shadows, or changes in lighting can impair performance and reduce the accuracy of identification.
- **Static Backgrounds:** The approach is mostly designed for circumstances with minimal dynamic change and rather static backgrounds.

Limitations

- **Complex Occlusions:** Highly opaque objects with few discernible features may be difficult for the system to comprehend.
- **Real-Time Constraints:** Real-time performance on edge devices is limited by hardware constraints.
- **Complex Backgrounds:** In scenarios with extremely complex or dynamic backdrops, performance may suffer.
- **CPU Performance Variability:** CPU specifications might affect inference performance, with lower-end CPUs possibly falling below the 10 FPS threshold.
- **Fixed Input Size:** The model might not generalize well to all image resolutions because it was trained on fixed input sizes.

Github Link: https://github.com/KanishkaRajendran/ObjectDetection

Colab Link:

https://colab.research.google.com/drive/1AeZf-GE1jzEYOv6pLNu_CYXGOOjDN4Uc