SafetyWhat AI Solutions Pvt.

Report on Object Detection System:

Inference Speed Results, System Architecture, and Optimization Strategies

Prepared by: Rohan Lohani

Date: 10-01-2025

1. Introduction

This report documents the inference speed results, system architecture, and optimization strategies used in the object detection system based on YOLOv5. The system is designed to detect objects and their sub-objects within images and videos, with the ability to crop and save sub-objects separately.

2. System Architecture

The system architecture is composed of the following components:

- **YOLOv5 Model:** A pre-trained YOLOv5s model loaded from the Ultralytics repository via PyTorch Hub. This model is used for object detection.
- **Image Processing Module:** Utilizes OpenCV to read images, perform bounding box operations, and visualize detections.
- **Sub-object Detection:** A custom function to detect and crop sub-objects within the bounding boxes of detected objects.
- **Output Module:** Handles saving the cropped sub-objects as images and exporting detection results in JSON format.

System Flow:

1. Load the YOLOv5 model.
2. Read and process an input image or video frame.
3. Detect objects using the YOLOv5 model.
4. Detect sub-objects within the detected objects.
5. Save the cropped sub-objects and generate a JSON output with detection details.

3. Inference Speed Results

The inference speed was tested on a video file to measure the system's performance. The following results were obtained:

- **Total Frames Processed:** 1200 frames
- **Total Time Taken:** 40.5 seconds
- **Average FPS (Frames Per Second):** 29.63 FPS

The inference speed indicates the system's ability to process almost 30 frames per second, which is suitable for real-time applications.

4. Optimization Strategies

To optimize the system, several strategies were employed:

- **Model Selection:** Using the YOLOv5s model, which is the smallest and fastest variant of YOLOv5, suitable for systems with limited computational resources.
- **Batch Processing:** Processing frames in batches to utilize GPU efficiently (if available).
- **Efficient Image Resizing:** Resizing images to the optimal input size for YOLOv5 to balance detection accuracy and speed.
- **Asynchronous Processing:** Employing asynchronous video capture and frame processing to minimize idle time between frame reads.

5. Conclusion

The object detection system demonstrates robust performance with an average processing speed of nearly 30 FPS, making it suitable for real-time applications. The system's architecture allows for the flexible handling of both objects and sub-objects, and the implemented optimizations ensure efficient processing. Further improvements could involve fine-tuning the model or employing hardware accelerations such as TensorRT for faster inference.