Report: Object and Sub-Object Detection System

Github Repository: https://github.com/jayxdev/sub-obj-detector

1. Introduction

The objective of this project was to develop a robust computer vision system capable of detecting objects and their associated sub-objects in a hierarchical structure. The system is designed to detect various objects (e.g., “person”, “car”) and their corresponding sub-objects (e.g., “helmet”, “tire”) in video streams. The solution must provide accurate and adaptable detection while maintaining real-time performance on a CPU. It also includes a functionality for retrieving cropped images of specific sub-objects for further analysis.

2. System Overview

The computer vision system leverages a fine-tuned YOLOv8n model to perform real-time object and sub-object detection. The YOLO model is trained to recognize different objects, and boundaries are used to identify sub-objects within those main objects. The system processes video frames in real-time and outputs detection results in a specified hierarchical JSON format. Furthermore, it supports the retrieval of sub-object images by cropping the identified regions from the video frames.

3. Object and Sub-Object Detection

- Main Object Detection: The system detects various objects such as “person”, “car”, “dog”, etc., in each frame of the video stream.

- Sub-Object Detection: Once the main objects are detected, sub-objects (e.g., head for persons, tires for cars, persons inside bus) are identified within the bounding boxes of the parent objects.

- Hierarchical Structure: The sub-objects are linked to their parent objects based on their relative positions. The system uses bounding box relationships to associate sub-objects with parent objects, creating a hierarchy where each object is uniquely indexed.

Example:

- For a detected person, the system will check for sub-objects such as “helmet” within the person's bounding box. The sub-object’s bounding box will be captured, and the hierarchical relationship will be saved.

4. JSON Output Format

The detection results are structured in the following hierarchical JSON format:

```json
{
  "object": "Person",
  "id": 1,
  "bbox": [x1, y1, x2, y2],
  "subobject": {
    "object": "Helmet",
    "id": 1,
    "bbox": [x1, y1, x2, y2]
  }
}
```

- **object:** Name of the main object.
- **id:** Unique identifier for the detected object.
- **bbox:** Bounding box coordinates for the object in the format [x1, y1, x2, y2].
- **subobject:** A nested dictionary for each sub-object associated with the parent object, including the sub-object's name, ID, and bounding box.

5. Sub-Object Image Retrieval

- The system allows users to retrieve cropped images of specific sub-objects. For example, after detecting a person, users can extract an image of the helmet (sub-object) associated with that person.

- The system ensures accurate image retrieval by referencing the hierarchical relationships between objects and sub-objects. The bounding box coordinates are used to crop the frame and save sub-object images.

Example: For the "Person" object, the system can crop and save the image of the “Helmet” sub-object if it is detected.

6. Inference Speed Optimization

- Optimization: The system is optimized to process video inputs at 10–30 frames per second (FPS) on a CPU. This ensures that the system meets real-time performance requirements while balancing detection accuracy.

- Benchmarking: The system was tested on a sample video, and the inference speed was measured. The average FPS achieved was 20 FPS, which satisfies the performance requirements for real-time video processing.

Optimization Strategies:

- Model size and inference speed were improved by fine-tuning the YOLOv8n model, optimizing the bounding box determination for sub-object detection, and using batch processing for video frames.

- The system utilizes efficient frame resizing techniques to reduce the computational load while maintaining detection accuracy.

7. Modularity and Extensibility

The design of the system is modular, allowing for the easy addition of new object-sub-object pairs (e.g., adding new classes such as “Bike” and “Handlebars”). This modular approach ensures that the system can be extended to handle different object detection scenarios.

- Instructions for Adaptation: To adapt the system to other object detection scenarios, users can update the object detection classes and the corresponding sub-objects in the configuration files. The detection model can be retrained with new classes if necessary.

8. Challenges and Limitations

- Occlusion Handling: The system handles occlusion scenarios by ensuring the object and sub-object detections are refined with non-maximal suppression to filter overlapping bounding boxes. However, in cases of severe occlusion, detection accuracy may decrease.

- Object Sub-Object Relationships: The accuracy of sub-object detection largely depends on the robustness of the YOLO model and the clarity of the bounding boxes. Any discrepancies in bounding box determination can affect the sub-object extraction process.

9. Benchmarking Results

- Test Video: A test video of traffic scenes was used to benchmark the performance of the system.

- Inference Speed: On a CPU, the system achieved an average of 20 FPS for the test video, meeting the performance requirement of 10–30 FPS.

- Detection Accuracy: The system demonstrated high detection accuracy with an average confidence score of 0.5 for main objects and sub-objects.

10. Conclusion

The system successfully meets the objectives of object and sub-object detection with hierarchical relationships, real-time video processing, and sub-object image retrieval. The modular design ensures that the system can be easily extended to handle additional object-sub-object pairs. With performance optimizations in place, the system operates efficiently on a CPU, ensuring scalability for edge devices.

Deliverables

1. Codebase: The full source code for the object detection system is available in the provided GitHub repository.

2. JSON Outputs: Sample JSON outputs have been provided for the test video, demonstrating the hierarchical structure of detected objects and sub-objects.

3. Benchmarking Report: The benchmarking results, including FPS and detection accuracy, are documented above.

4. Demo: An optional demo video showcasing real-time object and sub-object detection is included.Detected Objects are marked green and sub-object in blue.