**Concise Report: Real-time Object Detection and Tracking**

# **Introduction:**

I was tasked with developing a system capable of real-time object detection and tracking within a video stream. To achieve this, I leveraged YOLOv8 with a focus on the "yolov8n-pose" model for optimized real-time performance.

# **Challenges Encountered:**

1. **Time Constraints**: Given the 48-hour time frame, I opted to utilize a pre-trained model to expedite development and testing.

2. **Model Selection**: I explored various models, including Media Pipe and different YOLO versions. Ultimately, YOLOv8 was chosen for its balance of speed and accuracy.

3. **Integration Challenges**: Initially, I attempted to implement a Flask-based web application to display the output. However, this approach proved slow for real-time processing.

4. **Model Comparison**: I compared the "yolov8n-pose" model's performance to a standard YOLO model. While the standard YOLO achieved higher maximum FPS, the "yolov8n-pose" model offered competitive performance and superior accuracy.

# **Approach:**

I implemented the following approach:

1. Loaded the YOLOv8 model, specifically "yolov8n-pose" for its balanced performance.
2. Opened a video stream for real-time input.
3. Processed each frame using YOLOv8 for object detection.
4. Calculated FPS for each frame and stored the values.
5. Visualized the detection results on the frame and displayed it.
6. Utilized a predefined limit of 300 frames for real-time processing.
7. Generated a graph illustrating FPS over time for performance evaluation.

# **Results:**

Using the "yolov8n-pose" model, I achieved the following results:

* Average FPS: 22.30
* Maximum FPS: 33.79
* Minimum FPS: 0.13

# **Conclusion:**

The implemented solution leverages YOLOv8 with the "yolov8n-pose" model to achieve real-time object detection and tracking. It met the task's objectives, including detection and tracking accuracy, real-time processing speed, and model documentation. The "yolov8n-pose" model, although slightly slower than other models, demonstrated excellent accuracy and was well-suited for the given task.

# **Deliverables:**

The deliverables for this task include:

- A GitHub repository link, for code.

- A concise report summarizing the approach and results.

- A short video demonstrating the system in action.