Re-Identification in a Single Feed

Task:

Identification of each player ensuring same player retains the same ID even after going out of view in a 15-second video.

Given:

Object detection model – Fined-tuned Ultralytics YOLOv11

Approach and Methodology:

Initial Research and Setup

I began by thoroughly reviewing the Ultralytics documentation, focusing on two key areas: the QuickStart guide and Object Detection with Tracking capabilities. This provided the foundational understanding needed to implement player tracking effectively.

Model Testing and Validation

To understand the model's capabilities, I conducted initial testing on single-frame images to analyze the output format and detection accuracy. I then explored the model's object detection range to identify what types of objects it could reliably detect and classify.

Task-Specific Optimization

Since the objective was specifically to detect and track players, I configured the model to filter outputs and restrict detection classes to players only. This optimization reduces computational overhead and eliminates irrelevant detections.

Performance Enhancement Strategy

Initial Challenge: The basic Ultralytics installation, while functional, was limited to CPU-only processing, resulting in significantly slow real-time tracking performance that was unsuitable for live applications.

Solution: To achieve real-time performance, I enabled GPU acceleration by:

- Installing PyTorch separately with CUDA support
- Configuring the code to leverage GPU processing power
- This approach dramatically improved inference speed for real-time tracking scenarios

Tracking Algorithm Selection

Ultralytics YOLO offers two primary tracking algorithms, each with distinct trade-offs:

- **BoT-SORT:** Higher accuracy but slower processing speed
- **ByteTrack:** Faster processing but reduced accuracy

Final Implementation

The optimal solution combines:

- **Model:** Ultralytics YOLOv11 with GPU acceleration enabled
- **Tracker:** BoT-SORT algorithm for maximum accuracy
- **Confidence Threshold:** 0.8 (filters out low-confidence detections)
- **IoU Threshold:** 0.7 (optimizes object overlap detection)

This configuration balances processing speed with detection accuracy, making it suitable for real-time player tracking applications while maintaining reliable performance.