Re-Identification in a Single Feed

INTRODUCTION :

I chose the second assignment from the given options, which focuses on player tracking and re-identification in a single video feed. The goal of this task is to consistently identify and follow individual players throughout the video, even as they move across the field

APPROACH AND METHODOLOGY:

To solve the problem of tracking players and re-identifying them across video frames, I followed a structured approach using detection and tracking models:

- **YOLOv11 (Fine-Tuned)**
  I started by fine-tuning a YOLOv11 model to detect players, goalkeepers, and referees in each frame.
  YOLO (You Only Look Once) is a fast and accurate object detection model that gave me bounding boxes, class labels, and confidence scores for each person in the frame.

- **StrongSORT Tracker**
  After detecting the players, I passed the outputs to StrongSORT, a robust tracking algorithm. It connects detections across frames and assigns a unique ID to each player, keeping the ID consistent as long as the person stays visible.

- StrongSORT combines multiple techniques to improve tracking accuracy — it uses a Kalman Filter to predict the movement of each player, nearest neighbor matching for associating detections with existing tracks, and appearance features from a re-identification model to handle cases where players are close together or overlapping. This makes the tracking more stable and reliable throughout the video.

- **OSNet Re-ID Model**
  For extracting appearance features, I used OSNet (Omni-Scale Network), which is trained for person re-identification.
  It captured important visual details like clothing color and patterns, helping the tracker decide whether a person seen in the current frame is the same one from earlier frames.

TECHNIQUES TRIED AND THEIR OUTCOMESl

- For the assignment, I started by reading **a lot of research articles, blogs, and GitHub repositories** to understand different tracking techniques and the common challenges faced in **player re-identification in sports scenarios**. This helped me gain a better understanding of how different trackers work, especially in fast-paced and crowded environments.

• I began by experimenting with **YOLOv8 + Deep SORT**. This combination was simple to implement and worked fairly well in the beginning. However, I observed that **ID switching was quite frequent**, especially when players moved quickly or came close to each other. The tracker had difficulty maintaining the same ID across frames.

• To improve real-time tracking, I next used **YOLOv8 with the deep_sort_realtime library**. It offered better speed and smoother integration, but the issue of **ID instability** remained. Players still lost their IDs or got assigned new ones when they **overlapped** or **changed direction quickly**.

• Even after these improvements, ID switching still occurred, especially during partial occlusions or slight movements. To address this, I decided to improve appearance-based matching using a custom embedding model, like those available in deep_sort_pytorch, which are trained to extract strong visual features from person images.

• I also experimented with **classic Deep SORT** using the **mars-small128.pb encoder** (TensorFlow-based). This version produced **more stable and consistent tracking IDs** because it used **appearance features** along with motion. However, it still had some limitations in **crowded sports scenes**, where players looked similar or were tightly grouped.

• After trying all these methods, I found that the best performance came from combining **YOLOv11 + StrongSORT + OSNet**. This approach gave me the most **stable**, **accurate**, and **visually better tracking results**, even when players moved quickly, overlapped, or were partially occluded.

CHALLENGES FACED

• Initially, **Deep SORT** didn't maintain consistent IDs because it relied heavily on **motion-based tracking**. During rapid player movement, this caused **frequent ID switching**.

• When a player **reappeared after being occluded or out of frame**, Deep SORT often **assigned a new ID**, which broke continuity in tracking.

• Even after switching to the **OSNet model** with StrongSORT for appearance-based tracking, I observed **slight ID switches when players overlapped** or moved very close to each other.

• Tuning the **thresholds** (like IoU, max distance, and confidence scores) required **multiple rounds of trial and error** to reduce both **false positives** and **missed detections**.

• Integrating and experimenting with different trackers like Deep SORT (PyTorch/TensorFlow), deep_sort_realtime, and StrongSORT led to **compatibility issues** and required **custom adjustments** for formats and outputs.

CONCLUSION

The project is **not incomplete**, but with **more time and resources**, I believe it could be improved further. Specifically:

- The **YOLOv11 model**, although fine-tuned, sometimes detected **partial players (like half a body)** as full detections, even with high confidence. With access to **a more optimized YOLO model or additional training data**, this issue could be minimized, improving overall detection accuracy.
- With **more computational resources**, I could experiment with **larger or custom-trained OSNet variants** to enhance re-identification performance, especially in crowded scenes.
- A bit of **expert guidance or code reviews** would also help me **optimize the tracking pipeline** better — particularly in terms of reducing ID switches and refining threshold tuning.

Overall, the current solution works effectively, but I see clear potential for **refinement and performance gains** with more advanced tools and insights.