Objective:

Given a 15-second video (15sec_input_720p.mp4), identify each player and ensure that players who go out of frame and reappear are assigned the same identity as before.

Instructions:

- Use the provided object detection model to detect players throughout the clip.
- Assign player IDs based on the initial few seconds.
- Maintain the same ID for players when they re-enter the frame later in the video (e.g., near the goal event).
- Your solution should simulate real-time re-identification and player tracking.

Intro: Hi, I'm Sagar Rajak, a third-year BE student at VESIT Mumbai, pursuing Artificial Intelligence and Data Science with a current CGPA of 9.2..My major interests and work have been around machine learning, generative AI, and backend development..I was a finalist in SIH Hackathon 2024, and I interned at Kinzy Club Pvt Ltd (a parenting app) as a backend engineer with GenAI integration.I'm also a Senior Technical Officer at my college's AI Club, where I taught students: Machine learning (supervised and unsupervised algorithms),Computer vision (OpenCV workshop),Deep learning concepts (ANN, RNN, CNN)..Not in extreme depth, but enough to help them start their journey.

My solution: **Github** **bytetrack-video** **deepsort-video** **strongsort-video**

1. **My understanding of the problem**

   Initially, I thought tracking players in a short 15-second video would be easy—just detect them frame by frame. And i will save those id in some list or db then i can match them and identify …But then I realized the real challenge was ensuring that if a player left the frame and came back later they should keep the **same ID**,it is not easy to read frame and annotate easily ,. That's crucial for any real-world sports analytics application, where consistency of identity over time matters for stats and tactics analysis.

2. My approach and reasoning

Once I understood the problem better, I decided to solve it but as I solved I learnt more methods and techniques and to build a perfect solution I ended up doing three techniques ... .and tested three different tracking methods to see which one best handled re-identification when players went out of frame and reappeared….

(a) ByteTrack

I started with ByteTrack because it's a popular, lightweight tracker using IoU matching and Kalman filters. I liked its simplicity and speed. It maintains lost tracks for a buffer of frames, which helps with short occlusions.

However, I realized ByteTrack relies only on spatial overlap. It doesn't use appearance features, so if two players look alike or if occlusion is long, IDs can switch. So while it worked okay, it wasn't perfect for my problem.

(b) YOLO + DeepSORT

Next, I used DeepSORT because I read about it in blogs and its GitHub docs. They explained how it extends SORT by adding deep appearance features. That clicked for me—appearance-based re-ID is exactly what I needed for players leaving and re-entering…..At first I didn't understand how the internal parameters worked, like max_age, n_init, and cosine distance. So I went back to the docs and realized max_age controls how long tracks survive when detections are missing—super important for reappearance…I experimented a lot and hyper-tuned these parameters: I set max_age to 250 (10 seconds) to handle long occlusions, made appearance matching stricter (cosine distance 0.25), and required 4 initial detections for stability.

This approach gave me perfect stability in my tests, with 0 transient IDs, which really solved the problem.

(c) YOLO + StrongSORT

Finally, I tried StrongSORT after learning about it on forums and papers. It uses OSNet for appearance features, which is even more robust."

I adjusted its parameters as well: max_age 300 for even longer occlusion handling, cosine distance 0.2 for stricter appearance matching.

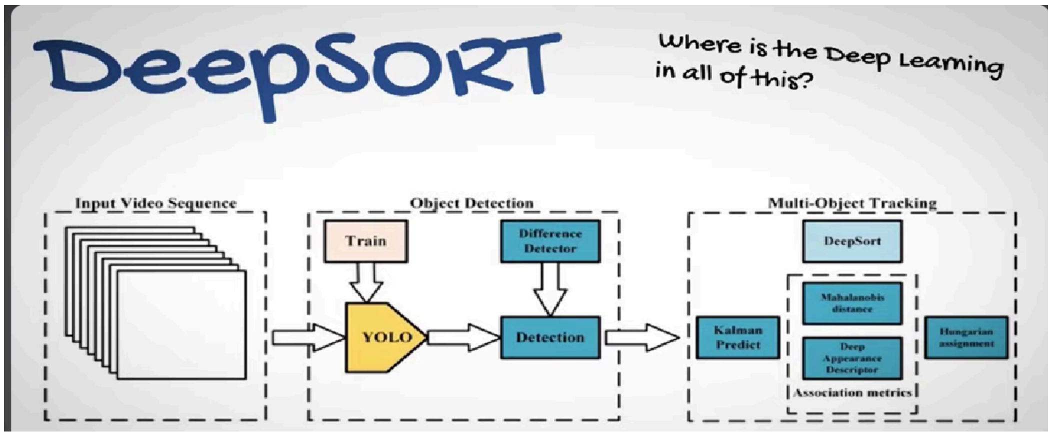
This gave me excellent re-ID even under tough conditions—only one transient false positive in my tests. To be honest i tweaked parameter multiple time to get proper ans still i was not able to find perfect solution…..

3. How YOLO, DeepSORT, ByteTrack docs helped me

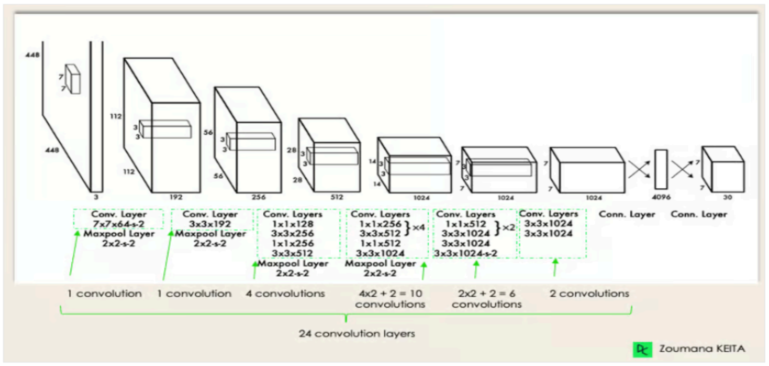
Honestly, I didn't understand YOLO or DeepSORT fully at first.i only had knowledge about open cv and cnn algo …… The YOLO documentation helped me understand the detection thresholds, NMS, and how to balance false positives and misses…….Reading DeepSORT's repo and blog posts taught me the whole architecture—how it uses embeddings, cosine distance, Kalman filters, and cascade matching."

ByteTrack's papers and repos helped me see its two-stage association strategy and the role of the lost track buffer.

These resources were essential. I went from treating them like black boxes to understanding how their internals worked so I could tune them.



YOLO architecture is similar to GoogleNet. As illustrated below, it has 24 convolutional layers, four max-pooling layers, and two fully connected layers.


448
448
7
7
112
112
3
192
56
56
256
28
28
512
14
14
1024
7
7
1024
7
7
1024
4096
30
Conv. Layer 7x7x64-s-2 Maxpool Layer 2x2-s-2
Conv. Layer 3x3x192 Maxpool Layer 2x2-s-2
Conv. Layers 1x1x128 3x3x256 1x1x256 3x3x512 Maxpool Layer 2x2-s-2
Conv. Layers 1x1x256 3x3x512 }x4 1x1x512 3x3x1024 Maxpool Layer 2x2-s-2
Conv. Layers 1x1x512 3x3x1024 }x2 3x3x1024 3x3x1024-s-2
Conv. Layers 3x3x1024 3x3x1024
Conn. Layer
Conn. Layer
1 convolution
1 convolution
4 convolutions
4x2 + 2 = 10 convolutions
2x2 + 2 = 6 convolutions
2 convolutions
24 convolution layers
Zoumana KEITA

YOLO Architecture from the original paper (Modified by Author)

Repo helped me: mikel-brostrom-github

4.Problems I faced and how I overcame them

One big problem was **false positives—players** detected for just a few frames. I added filtering based on area, aspect ratio, and NMS thresholds to reduce noise……Another was **IDs switching** when players reappeared some time id used to change very fast later i understood it is called **fragmentation problem i faced this in my deep sort algo** . Initially, my ByteTrack config lost them too soon, so I increased the lost buffer. But even then, for longer occlusions, ByteTrack wasn't enough—so I switched to DeepSORT and StrongSORT for appearance-based matching.

Finally, I faced parameter sensitivity. Changing thresholds too aggressively broke tracking. I solved it by systematically logging results, checking frame coverage, and finding the sweet spots for parameters.

5.How I hyper-tuned my models:

I read about them in deep for deep-sort and strong-sort algo..

IOU: Measures overlap between two boxes to see how well they match.

NMS: Removes overlapping detections by keeping only the highest-confidence ones.

Cosine Similarity: Compares appearance features to tell if two detections look alike.

Kalman Filter: Predicts the next position of a tracked object based on motion.

Hungarian Distance: Finds the best assignment between predicted tracks and new detections to minimize overall cost.

After understanding the role of each parameter, I deliberately tuned them:

- **YOLO conf threshold**: Increased to 0.85–0.87 for fewer false positives
- **max_age**: Extended to 250–300 to handle long occlusions
- **n_init**: Required multiple consecutive detections to avoid noise tracks
- **Cosine distance**: Made stricter (0.25 or 0.2) to prevent ID confusion
- **Aspect ratio and area filters**: Applied to ensure detections are human-like.

nn_budget: These vectors are compared with the embedding of new detections.
A higher nn_budget means more history is preserved to match reappearing players.

6. How I plan to improve it further:

First, I thought about training a **custom re-ID model** on my own dataset, with players from my target league or team. That way, the appearance embeddings would be even better at distinguishing players……Second, I want to add **team identification** by color clustering. Right now, I’m just tracking individuals. By adding a color-based team ID, I could annotate players as Team A or Team B automatically.

Third, I plan to experiment with even stronger embedders or hybrid approaches, like combining OSNet with temporal consistency constraints…. And to be honest i'm not extreme expert so i need to read more about it right now ..

7.**Brief Note on Relevant Experience:**

I have worked on a computer vision project involving tennis player tracking, where I used YOLO for detection and applied tracking algorithms .This gave me hands-on experience with object detection, re-identification, and frame-by-frame video analysis….Additionally, I have assisted in teaching OpenCV to students at my college, helping them understand core concepts such as image processing, feature detection, and practical applications of computer vision techniques.

Im not expert but yes i do know things …..

I truly loved solving this assignment sadly i couldn't solve it with proper accuracy but **yes i learnt more about cnn, open cv , object tracking ,deepsort, botsort, strongsort and** definitely this going to help me in future ..so yes it was good experience…