Player Re-Identification in Sports Footage

Name: Tejaswini G C
Role: AI Intern
Company: Liat.ai
Date: June 30, 2025

1. Introduction

This report details my work on the Liat.ai AI Intern assignment: Player Re-Identification in Sports Footage. I focused on Task Option 2, aiming to track players in 15sec_input_720p.mp4 and keep their IDs consistent, even after they went off-screen. This was a valuable learning experience.

2. My Approach

My strategy involved two steps: detecting players and then consistently tracking them.

- Tracking Algorithm Choice: I initially experimented with Norfair and DeepSORT, but these often produced duplicate bounding boxes and failed to re-identify players returning to the frame. Based on online research suggesting its effectiveness, I switched to ByteTrack.
- Model Used*:* I utilized the YOLOX model with the bytetrack_x_mot17.pth.tar checkpoint, as it's specifically trained for person tracking and compatible with ByteTrack. I tried to use the given model best.pt. However, this caused several problems during the model loading phase and ultimately, it was incompatible with the ByteTrack framework as it was set up.

- Key Settings: I adjusted parameters like --conf, --track_thresh, --match_thresh, --track_buffer, --min_box_area, and used --device cpu due to my system's limitations.

3. Challenges and What I Learned

Getting the system running involved significant troubleshooting:

- Installation Issues: I faced initial difficulties with dependencies like onnxruntime and simplifier, requiring careful pip and conda command management.
- Model Loading Problems:
    - A PermissionError occurred when loading the bytetrack_x_mot17.pth.tar file. This was resolved by restarting my computer to clear file locks.
    - I also had to modify tools/demo_track.py to correctly load the model's weights (ckpt["model"] and weights_only=False) because its internal structure didn't directly match the script's default expectations.

- Code Compatibility Errors:
    - I encountered AttributeError: module 'numpy' has no attribute 'float' due to NumPy version changes. I fixed this by replacing np.float with float or np.float64 in yolox/tracker/matching.py and yolox/tracker/byte_tracker.py.

- Runtime Warnings: An OMP: Error #15 warning was resolved by setting $env:KMP_DUPLICATE_LIB_OK="TRUE".

Despite these hurdles, the system successfully loads the model, detects players with high confidence and performs basic tracking.

4. Remaining Challenges & Future Steps

While basic tracking works, robust player re-identification is still difficult:

- ID Swaps: Players crossing paths often cause their IDs to switch.
- New IDs on Re-entry: Players returning to the frame after a period away get new IDs instead of their old ones.

With more time, I would:

- Integrate Appearance Features: Implement methods like Deep SORT or dedicated re-identification models to use a player's visual appearance for more reliable re-identification, especially after long occlusions or re-entry.
- Systematic Parameter Tuning: Conduct more thorough testing of tracking parameters (--track_thresh, --match_thresh, --track_buffer) to find optimal settings.
- Utilize GPU: Switching to GPU processing would significantly speed up development and allow for higher-quality video analysis.

5. Conclusion

This assignment was a fantastic learning experience. I successfully got a complex player tracking system working, dealing with many real-world technical issues. It was very satisfying to see the system detect players.

While fully perfect player re-identification is a big challenge, I have learned a lot about tracking systems, how important debugging is and the exciting potential of computer vision in sports. I'm eager to learn more in this field.