Football Broadcast Player Detection and Re-Identification using YOLOv11

Final Project Report

Arpit Chauhan

1 Objective

The objective of this project is to design a robust, real-time system capable of detecting and consistently tracking football players in broadcast videos. A primary challenge addressed is ensuring that each player maintains a unique and persistent identity across video frames—even when temporarily occluded or out of view due to camera motion or scene transitions.

2 Approach and Methodology

The following steps were undertaken to build the Football Player Detection and Re-Identification System:

1. **Frame Extraction:** A 15-second football broadcast video was used as the input source. A total of 75 frames were extracted using OpenCV for annotation and training.
2. **Dataset Creation using Roboflow:** Each frame was uploaded to Roboflow, where manual annotations were performed. The labeled categories included:
   - Football players
   - Referee
   - Football
   - Other objects (e.g., goalposts, etc.)

   Data augmentation techniques such as flipping, brightness/contrast adjustment, and blurring were applied. The final dataset contained:
   - 238 images total
   - 222 images for training
   - 11 images for validation
   - 5 images for testing

   The dataset is publicly available on Roboflow at: https://app.roboflow.com/arpittchauhann/de vpmbm/4

3. **Model Selection and Training:** The YOLOv11 model was selected for object detection. The training process was carried out using the Jupyter Notebook file `Football_Analysis_System (1).ipynb`, where the model was trained for 150 epochs with early stopping applied. The best model checkpoint was saved as `best.pt`.

4. **Post-Training Inference:** Two inference pipelines were implemented:

   - `yolo_inference.py` – for basic object detection using pretrained weights
   - `main.py` – for detection and player tracking using the custom-trained model

5. **Re-Identification and Output:** The output included annotated videos with consistent player IDs, and tracking metadata was saved as a `.pkl` file inside the `tracker_stubs/` folder.

3 Techniques Tried and Outcomes

- **Pretrained YOLOv11 Inference:** Initially, the `yolo_inference.py` script was used to perform basic object detection using pretrained YOLOv11 weights. The model was able to detect generic classes (players, ball) but without persistent IDs or tracking across frames.

- **Custom YOLOv11 Training and Re-ID:** A custom dataset was prepared and annotated using Roboflow. The YOLOv11 model was trained for 150 epochs. The training achieved a final mAP@50 of **0.888** and mAP@50–95 of **0.429**, with high precision and recall across classes like `football players`, `referee`, and `football`.

- **Custom Tracking with Re-Identification:** The `main.py` script implemented a full pipeline using a custom tracker. The model was initialized with the fine-tuned weights, and the player tracking results were written to `tracker_stubs/player_detection.pkl`. Visual annotations were drawn across all video frames, and a final output video was saved in `output_videos/`.

4 Challenges Encountered

- **Re-ID Gaps After Frame Exit:** The model struggled to reassign the same identity to players who left the field of view and returned after a few seconds, leading to fragmented tracking.

- **Visual Similarity Confusion:** Players with nearly identical uniforms, especially teammates, caused the model to incorrectly swap or duplicate identities.

- **Blurred or Occluded Frames:** Fast movements or partial occlusions led to degraded visual features, making identity re-matching highly unreliable.

5 Model Performance and Results

The YOLOv11 model was trained for a maximum of 150 epochs with early stopping enabled (patience = 100). Training was halted at epoch 148, with best performance at epoch 48.

Final Evaluation Metrics (Validation Set)

Category	Precision	Recall	mAP@50	mAP@50–95
Football	0.811	0.778	0.747	0.373
Football Players	0.984	0.922	0.953	0.469
Referee	0.918	0.929	0.963	0.445
Overall	0.904	0.876	0.888	0.429

Table 1: Per-Class Detection and Re-Identification Performance

Speed Performance

- Preprocessing Time: 0.2 ms/image
- Inference Time: 17.4 ms/image
- Postprocessing Time: 1.1 ms/image

```
                 all         11        152       0.87      0.883      0.868      0.373

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
    147/150      10.4G     0.8305     0.3913      1.048        156        640: 100% 14/14 [00:10<00:00,  1.39it/s]
                Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100% 1/1 [00:00<00:00,  4.31it/s]
                  all         11        152      0.872      0.883      0.864      0.371

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
    148/150      10.4G     0.8166     0.3901      1.045        170        640: 100% 14/14 [00:10<00:00,  1.38it/s]
                Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100% 1/1 [00:00<00:00,  4.69it/s]
                  all         11        152      0.876      0.878       0.86      0.364
EarlyStopping: Training stopped early as no improvement observed in last 100 epochs. Best results observed at epoch 48, best
model saved as best.pt.
To update EarlyStopping(patience=100) pass a new patience value, i.e. `patience=300` or use `patience=0` to disable EarlyStop
ping.

148 epochs completed in 0.492 hours.
Optimizer stripped from runs/detect/train2/weights/last.pt, 51.2MB
Optimizer stripped from runs/detect/train2/weights/best.pt, 51.2MB

Validating runs/detect/train2/weights/best.pt...
Ultralytics 8.3.160 🚀 Python-3.11.13 torch-2.6.0+cu124 CUDA:0 (Tesla T4, 15095MiB)
YOLO11l summary (fused): 190 layers, 25,282,396 parameters, 0 gradients, 86.6 GFLOPs
                Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100% 1/1 [00:00<00:00,  4.16it/s]
                  all         11        152      0.904      0.876      0.888      0.429
             football          9          9      0.811      0.778      0.747      0.373
     football players         11        131      0.984      0.922      0.953      0.469
              refree          7         12      0.918      0.929      0.963      0.445
Speed: 0.2ms preprocess, 17.4ms inference, 0.0ms loss, 1.1ms postprocess per image
Results saved to runs/detect/train2
💡 Learn more at https://docs.ultralytics.com/modes/train
```

Figure 1: YOLOv11 Training Summary: Loss curves, class-wise metrics, and evaluation output

6 Future Work and Possible Extensions

To enhance performance and robustness, especially in complex real-world settings, the following improvements can be considered:

- **Strengthen Re-Identification Across Gaps:** Train the system to better associate players who momentarily exit and re-enter the frame using more refined embeddings or temporal matching techniques.
- **Develop Player-Specific Re-ID Features:** Tailoring Re-ID by using features like jersey text, body build, or visual patches like logos could make the system more resilient to appearance overlap.
- **Apply Motion-Based Forecasting:** Motion prediction strategies such as Kalman filters or trajectory estimation can help maintain consistent identities during short disappearances.
- **Boost Efficiency for Real-Time Applications:** Convert and optimize the model using inference engines such as ONNX or TensorRT to reduce frame processing delays.
- **Scale to Multi-View Tracking:** Extend the architecture to handle multi-camera input and match players across views using cross-camera Re-ID techniques.

7 Conclusion

This project successfully demonstrated a working pipeline for football player detection and re-identification in broadcast videos using YOLOv11. The system showed significantly better performance when trained on a custom dataset and integrated with a tracking mechanism. Future improvements can include real-time deployment, jersey number OCR, and larger-scale datasets for robust generalization.