Soccer Player Tracking

Thomas Bornhorst
Case Western Reserve University
thb34@case.edu

Colin Chenoweth
Case Western Reserve University
cwc63@case.edu

Olivia Tchilibou
Case Western Reserve University
axt619@case.edu

Abstract

Player-tracking technology has become increasingly important in sports analytics, providing teams with valuable insights into player movements and strategies. This paper focuses on the problem of tracking the position of soccer players throughout a game using deep learning models. The Faster R-CNN model, an advancement in object detection, was employed for this purpose due to its ability to seamlessly integrate region proposal networks (RPNs) for end-to-end training. This integration allows for faster and more accurate detection compared to earlier approaches, setting a new standard in object detection.

We utilized the Soccer Track dataset, comprising 60 short videos of soccer games annotated with player positions and bounding box coordinates. Through preprocessing steps, the dataset was prepared for training, and a Faster R-CNN model was implemented using the PyTorch framework. The model was trained on a large number of frames and evaluated using various metrics.

Despite computational challenges, including long training times and GPU storage limitations, the model showed promise in tracking players across video frames. While precision in classifying player teams was lower than expected, the overall results demonstrated that this approach holds potential for advancing player tracking in sports. Future work will explore improvements in data processing, model training, and evaluation to achieve more accurate tracking and analysis.

1. Introduction

In recent years, sports teams have increasingly relied on data to improve performance. This shift began with baseball when the 2001 "Moneyball" Oakland Athletics used data in order to achieve unexpected success and has now spread to many other sports such as basketball and soccer [3]. Using advanced data analysis, teams can gain an advantage over their competitors.

This paper focuses on a key problem in soccer: tracking the position of all players during a game. Player-tracking technology helps teams understand how their players move and how to improve strategies as well as managing players' workloads, but traditional tracking systems can be expensive and complicated [2]. We're exploring a different way to track players using a deep learning model.

As outlined in [10], Faster R-CNN, a significant advancement in object detection, evolved from earlier models like R-CNN and Fast R-CNN. Unlike its predecessors, Faster R-CNN integrates the region proposal network (RPN) directly into the model, enabling end-to-end training and eliminating the need for external algorithms for region proposal generation. Its architecture consists of a backbone network to extract feature maps, an RPN to generate region proposals, and additional layers for classification and bounding box regression. By seamlessly integrating these components, Faster R-CNN significantly improves both speed and accuracy compared to previous approaches. This breakthrough has set new standards in object detection and has formed the basis for many subsequent models, marking a milestone in the evolution of deep learning-based object detection systems.

We used a dataset of 60 short soccer videos with annotations showing player positions. By using Faster R-CNN, we hope to make player tracking more accurate and extract useful information that can help soccer teams make better decisions.

1.1. Dataset

To train and evaluate our player-tracking model, we used the Soccer Track dataset, which consists of 60 short videos, each 30 seconds long, capturing soccer games from a top-down view [11]. The dataset includes approximately 54,000 frames and provides detailed annotations for each frame,

specifying the bounding box coordinates for each player and the ball.

Each video is annotated with a CSV file containing the precise coordinates for bounding boxes around players, enabling the identification of their movements throughout the game. The dataset also provides additional wide-angle views that were not used in our current study, as well as GNSS coordinates for each player.

Due to the dataset's size and complexity, we implemented preprocessing steps to ensure data quality and consistency. This involved extracting video frames, filtering out frames with missing annotations, and calculating bounding box coordinates. We split the dataset into training and test sets in a 90-10 ratio, respectively. This balanced split ensured robust model training and allowed for effective evaluation of the model's performance.

The dataset provides a solid foundation for training deep learning models for multi-object detection and tracking, offering realistic scenarios with multiple moving players and varying camera angles. This diversity makes it an ideal choice for developing robust player-tracking solutions.

2. Related Work

Multi-object detection and tracking is a deep field with a variety of different methods that have been used with numerous different applications. As all presented in [9], there are a multitude of different possible backbone networks to extract the features of an image before even getting to the more complicated nature of some of the networks. As partially discussed before, there are many versions of the R-CNN network, including more such as the R-FCN and Feature Pyramid Network structures. In comparison to the two-stage detectors of R-CNNs, there are also many one-stage detectors such as You Only Look Once (YOLO). YOLO has a multitude of different versions with each trying to improve on the last [4]. Multi-object tracking (MOT) is its own unique field as well with Deep Network MOT, End-to-end Deep Learning MOT, RNN-based MOT, and other methods for accomplishing the task [8].

For this paper, it's important to look at multi-object detection in sports contexts to see how past works have approached similar problems. In [6], assumptions about the sport were made in conjunction with a particle filter and other model aspects. In [11], YOLO v5 detection of players on a field is used with additional processing for object tracking, including the use of Kalman Filters. [1] shows impressive work of turning 2D pose-tracking of soccer players into 3D tracking of their body positions. [7] establishes a unique model for the tracking of basketball players. The paper utilizes a model that is constrained using the context and assumptions of the sport and then uses a random decision forest to process the work. [5] provides a survey of ball tracking across a variety of different sports with the use of a Kalman Filter, Particle Filter, Trajectory Based approach, and more. There are many papers that take a dive into some aspect of player detection or tracking in sports, each towards a new direction or with a different method.

3. Methods

We used Python and the packages PyTorch, TorchVision, Pandas, SciKit-Learn, and MatPlotLib to design our entire Soccer Player Tracking system. The bulk of our implementation was in the following six files: `prep_data.py`, `faster_RCNN.py`, `test.py`, `metrics.py`, `annotate_frames.py`, and `tracking.py`.

3.1. prep_data.py

This file had one main method `prep(...)` and numerous helper methods. `prep(...)` reads the numerous `.csv` and `.mp4` files from the dataset, extracts the frames into `.jpg` format (if the parameter `save_frames=True`), and puts all the annotations into one Pandas data frame with the additional column specifying which frame the annotations are for. This method also converts the bounding boxes from left, top, height, and width to left, top, right, bottom, which are the columns required by the FasterRCNN model we used.

3.2. faster_RCNN.py

This is the core file of our project. It converts the data frame from `prep_data.py` into a specific `SoccerTrackDataset`, sets up the FasterRCNN model, and trains it. When running the file, it has an `-l` argument flag that will load a model instead of training one from scratch.

3.2.1 SoccerTrackDatatset

In order to successfully train and predict SciKit-Learn's FasterRCNN model, we made our own Dataset class that ultimately returns the image as a Tensor and the target dictionary, which holds the ground truth bounding boxes, labels, and image_id. For simplicity sake and to hopefully improve performance, we decided to only use three labels: team 1, team 2, and ball.

3.2.2 FasterRCNN Model

For our FaserRCNN model, we used SciKit-Learn's version, specifically we used `faserrcnn_resnet50_fpn()` without any pre-trained weights. We made sure to correctly change the number of classes to fit our problem before training.

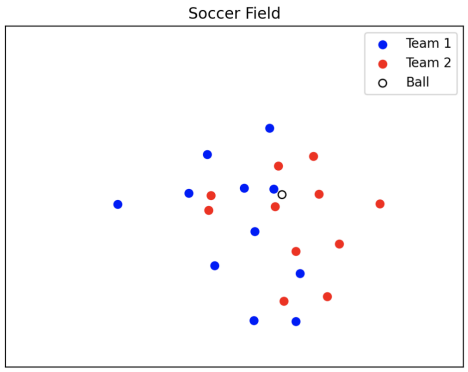

Figure 1. Screenshot of player animation from tracking.py.

3.2.3 Model Training

Because the dataset was so large, we decided to use a 90-10 random split for the train and test sets respectively. We made parameters for number of epochs and batch size that we could tune while training, and made it conducive to run on a GPU. The optimizer we used in training was the `torch.optim.SGD()` optimizer with a learning rate of 0.005, momentum of 0.9, and and weight decay of 0.005. Lastly, we set it to save every 50 batches to help with trouble shooting if needed.

3.3. test.py

The test file was primarily used to print predictions from a saved model to check performance during training, but became adapted to save predictions to a CSV file for future use of getting performance metrics, creating visuals, and tracking. There is one main method and few helpers methods, the main method can print predictions and the ground truth for frames, and can also add all the predictions to a Pandas data frame and save to a CSV file.

3.4. metrics.py

This uses the `.csv` files from `test.py` to output the average precision and recall for the model. The threshold for what overlap between a ground truth and predicted bounding box is considered a "true positive" is tunable, but by default its set at 0.5.

3.5. annotate_frames.py

In order to get a visual representation of how well predictions are, this file overlays the predicted bounding boxes from `test.py` onto the frames extracted with `prep_data.py` and saves them for future reference.

3.6. tracking.py

In order to get a better visual representation of the players and how the annotations could be used in a simple way, this file creates an animation with each player represented as a simple dot on a mock soccer field. Fig. 1 is a screenshot from that animation. This file also was used to test tracking possession by team throughout a game using annotations by checking which team the closest player to the ball was on at each frame. This is just a simple example of the metrics that could come from accurate tracking data.

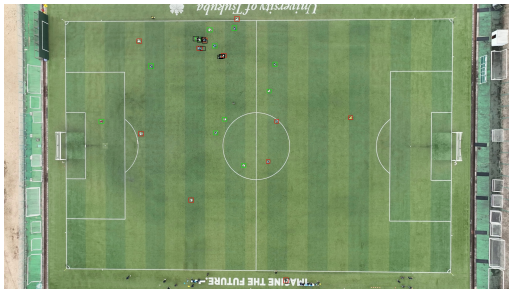

Figure 2. Top-down view frame 25199 annotated with model predictions.

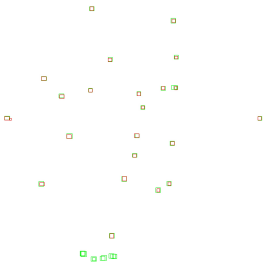

Figure 3. Comparison of true annotations boxes (red) versus predicted boxes (green).

4. Experiments

The model was trained for 16 epochs because looking at results while it was training, the model started to plateau at a loss value of 0.67 for epochs 14-16. The model was then run on the 10% of the data left for testing and annotations were created. First, the results were evaluated qualitatively by looking at the frames annotated with the predicted bounding boxes. These boxes were quite close to the ground truth boxes (as shown in Fig. 3) with predictions just having extra boxes, but the class labels were quite off (as shown in Fig. 2).

Next, to look at the results quantitatively, Tab. 1 shows the precision and recall for various intersection over the union (IOU) thresholds and whether classes were taken into account. There is a significant performance jump when classes are ignored. We believe relatively low performance of class labeling may be due to the similarity in the two teams kit colors (white versus blue and white), the goalies having different kit colors from the rest of the team, and the

IOU, C	0.5, T	0.25, T	0.5, F	0.25, F
Precision	0.1479	0.1749	0.2761	0.3102
Recall	0.4631	0.5474	0.8644	0.9712

Table 1. Performance metrics based on different IOU thresholds and whether class labels were accounted for (T) or not (F).

small top view only of the players, with almost all of the players having black hair.

Another interesting thing to note, is precision is much worse than recall for all the variations, but this can easily be explained by the model producing way too many bounding boxes, in particular too many ball boxes (as shown in Fig. 2). The reason we believe the model to produce an excess of bounding boxes is because in the frames there are a good number of people that not on the field such as substitutes, coaches, and referees, there are a number of balls not on the field but in view of the camera, and many of the players shoes are of a similar color as the ball and can easily be mistaken.

For comparison, metrics were also calculated on the testing data with 23 random annotations of appropriate player size per frame. For this randomized data with an IOU threshold of 0.25, the average precision and recall were 2.5e-05 and 2.5e-05. So as expected, our model performs significantly better than random boxes.

Because the predicted class labels were so inconsistent, we were not able to get additional information from our predictions using `tracking.py`, but when tested with the ground truth annotations we were quite happy with the results. The animation Fig. 1 seemed to be a very accurate representation of the game, and the possession metrics appear to be accurate from our own watching of the clips.

5. Next Steps

The two main problems with the model as it stands, is it produces too many bounding boxes and the labels of the boxes are incorrect. The former potentially would be the easiest to fix by simply cropping the frames so only the field is in view and setting a limit to the number of boxes that it can produce. To improve the performance of the predicted class labels, we could look into playing around with the contrast of the images, maybe find a new dataset with a greater visual difference between the kits, or maybe simply set a limit to the number of boxes of each individual label that the model can produce.

A big problem with the training of the model was that each epoch took so long train, partially because of how much data there was to process. If this could be sped up, such as through parallelization or only working on a much smaller set of the data, then more epochs could be run and hopefully the model could become much better. Part of the problems training the model came with computational limitations, specifically that the GPU storage limits greatly limited our batch sizes, and so with better computational power the model could possibly be trained more effectively.

The model itself could also be approached in a different way. Although there was trouble getting it to work for this paper, in the future somebody could try to use the SciKit-Learn Faster R-CNN model with pre-trained weights on this data to see if it would do any better, or at least train faster.

One possible cause of the issues with the model was that the players in the video were very small compared to the overall view. Future models could look into using other datasets with different camera angles and perspectives.

Future work could look more into what this data could actually be used for, beyond animations and possession tracking. The tracking data could be used to tally statistics like passing success rate per player, tackle success per player, possession by player, and other relatively simple statistics. As well, it could be used for more advanced metrics that would help determine the weaknesses of the team beyond what the average person could deduce from watching the game. As well, work onto tracking players between frames could be looked into and a naive approach of checking the closest player to the last frame might work quite well.

6. Conclusion

The Faster R-CNN model trained in this paper was able to successfully track soccer players in the SoccerTrack dataset, but it was not able to differentiate between teams so was not able to provide valuable insights into this world. Without class labels and an IOU threshold of 0.25, the Faster R-CNN model we trained had a precision of 0.31 and a recall of 0.97, while using class labels and the same IOU threshold those values are cut almost in half with a precision of 0.15 and recall of 0.46. Team tracking and possession tracking were successfully able to be set up (tested with the ground truth annotations), but it could not be used for this paper's model's predictions due to the inaccuracy of the classes. Future work will be able to expand on these pursuits and hopefully lead to a better solution which can successfully distinguish between two separate soccer teams, have no extraneous bounding boxes, and provide valuable insights to teams.

References

[1] Lewis Bridgeman, Marco Volino, Jean-Yves Guillemaut, and Adrian Hilton. Multi-person 3d pose estimation and tracking in sports. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 2487–2496, 2019. 2

[2] Martin Buchheit and Ben Michael Simpson. Player-tracking technology: Half-full or half-empty glass? *International*

Journal of Sports Physiology and Performance, 12:35–41, 2017. 1

[3] Tim A. Herberger and Christoph Litke. The impact of big data and sports analytics on professional football: A systematic literature review. In *Digitalization, Digital Transformation and Sustainability in the Global Economy*. Springer, 2021. 1

[4] Peiyuan Jiang, Daji Ergu, Fangyao Liu, Ying Cai, and Bo Ma. A review of yolo algorithm developments. *Procedia Computer Science*, 199:1066–1073, 2022. 2

[5] Paresh R. Kamble, Avinash G. Keskar, and Kishor M. Bhurchandi. Ball tracking in sports: a survey. *Artificial Intelligence Review*, 52:1655–1705, 2017. 2

[6] Matej Kristan, Janez Perš, Matej Perše, , and Stanislav Kovačič. Towards fast and efficient methods for tracking players in sports. In *Proceedings of the ECCV Workshop on Computer Vision Based Analysis in Sport Environments*, pages 14–25, 2006. 2

[7] Jingchen Liu, Peter Carr, Robert T. Collins, and Yanxi Liu. Tracking sports players with context-conditioned motion models. In *2013 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1830–1837, 2013. 2

[8] Wenhan Luo, Junliang Xing, Anton Milan, Xiaoqin Zhang, Wei Liu, and Tae-Kyun Kim. Multiple object tracking: A literature review. *Artificial Intelligence*, 293:1066–1073, 2021. 2

[9] Sankar K. Pal, Anima Pramanik, J. Maiti, and Pabitra Mitra. Deep learning in multi-object detection and tracking: state of the art. *Applied Intelligence*, 51:6400–6429, 2021. 2

[10] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *CoRR*, abs/1506.01497, 2015. 1

[11] Atom Scott, Ikuma Uchida, Masaki Onishi, Yoshinari Kameda, Kazuhiro Fukui, and Keisuke Fujii. Soccertrack: A dataset and tracking algorithm for soccer with fish-eye and drone videos. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3569–3579, 2022. 1, 2