FOOTBALL ANALYSIS SYSTEM WITH yoloV8

# Introduction:

## Project Overview:

## The Video dataset:

# Object Detection (YOLO) and tracking:

## Definitions and concepts:

**Object detection:** a fundamental task in computer vision, encompassing the identification and localization of objects within digital images or video sequences. This process involves assigning bounding boxes to specific objects within the scene, effectively drawing attention to their presence and spatial location.

**YOLO (You Only Look Once):** is a state-of-the-art deep learning algorithm specifically designed for real-time object detection. Unlike traditional methods that scan images sequentially, YOLO operates on the entire image simultaneously, enabling rapid and efficient object detection.

**The core principles of YOLO can be summarized as follows:**

* Image Partitioning: The input image is divided into a grid of cells.
* Object Prediction: Each cell within the grid is responsible for predicting the presence and location of objects within its boundaries. This involves predicting bounding box coordinates and assigning class labels to the detected objects.
* Confidence Scoring: YOLO assigns a confidence score to each prediction, quantifying the algorithm's certainty about the presence and classification of the object.

The unique advantage of YOLO lies in its ability to process the entire image in one pass, leading to significantly faster detection speeds compared to traditional methods. This makes YOLO particularly valuable in applications requiring real-time performance, such as autonomous driving, surveillance systems, and robotic vision.

**In essence, YOLO offers a highly efficient and robust approach to object detection, providing real-time performance while maintaining accuracy and precision. Its versatility and speed have propelled it to the forefront of computer vision applications.**

**Technical Advancements in YOLOv8:**

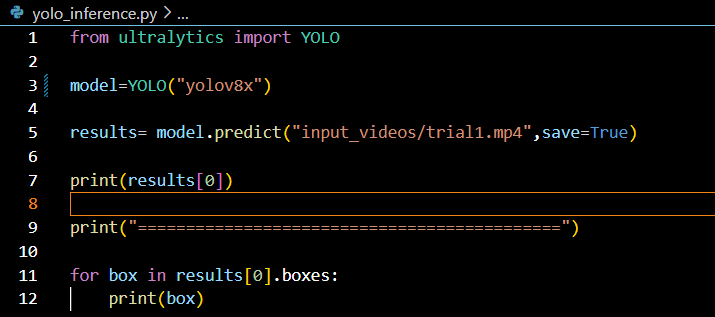
YOLOv8 enhances its predecessors by incorporating adaptive grid sizes and multi-scale detection capabilities, allowing it to effectively detect objects of various sizes across different resolutions. Improved convolutional layers, advanced activation functions, and optimized batch normalization contribute to its heightened accuracy and speed, it includes 80 pre-defined classes and 640 figSize.

## Basic YOLO use case:

You can use yolo with ease by using a library called **ultralytics.**

Install it on your environment first using the command %pip install ultralytics (on an ipython file) or !pip install ultralytics on the terminal.

As you can see in the code beneath, to use the yolo model from ultralytics we need to first import it from the library. Then we load the model using YOLO() you can choose which model you load (more documentation is on [Github](https://github.com/ultralytics/ultralytics)). In this demo we choose the model YOLOv8x that is trained on COCO which include 80 pretrained classes.



To make predictions we use the method model.predict and we give it as arguments the path to the video, and in case we want the prediction to be saved we add save=True (the default value is False). When you use save=True and you run it automatically creates a folder named runs in which you can find the result video.

The result is as shown in this figure:

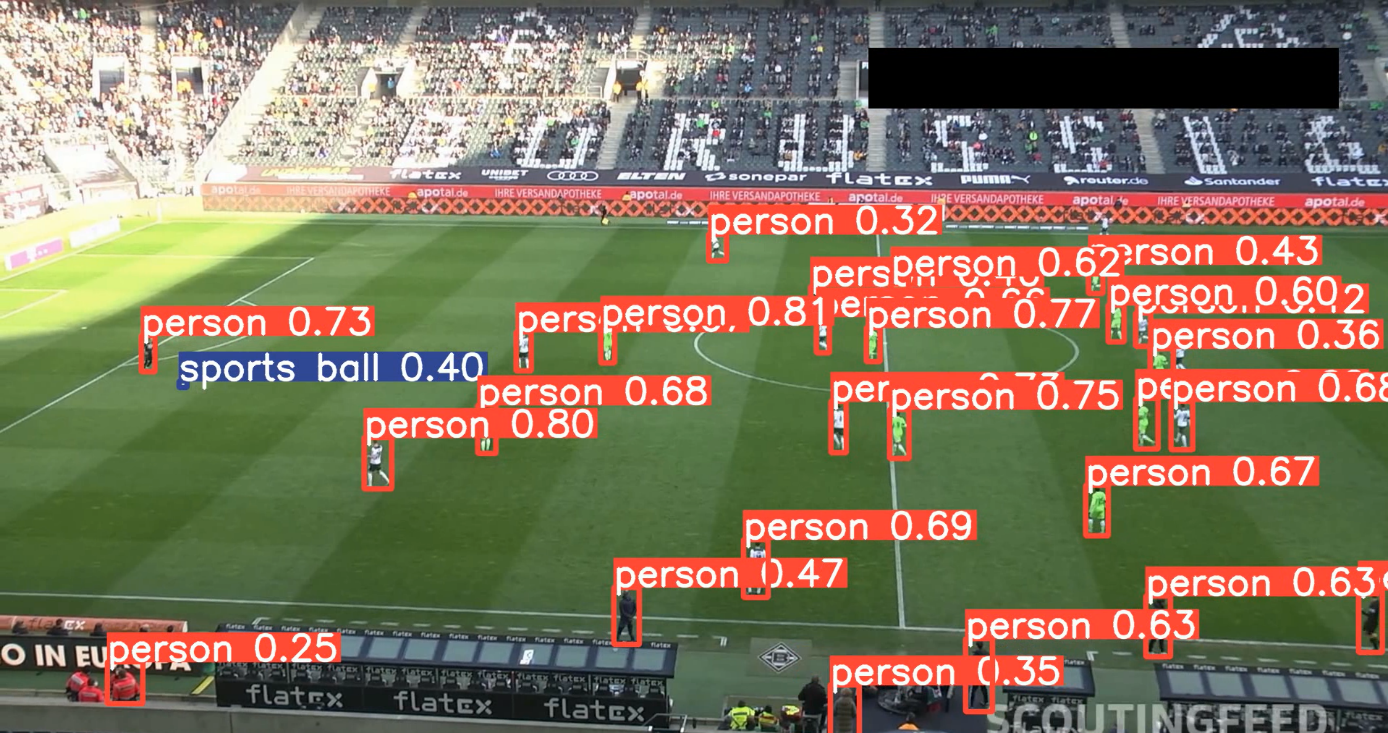


Figure 1: Frame of the result of the model prediction on the demo python inference file.

As you can see the model successfully detects the players and the referees but they are only detected as person, same for sports ball. The other problem that we have with using this model is that it’s detecting people off the pitch as well (such as coaches and staff) it would be nicer if they aren’t detected.

## Training and Dataset:

In this subsection, we introduce a training dataset developed to address significant challenges in detecting and tracking footballs during games. The existing model inconsistently detects the ball and mistakenly identifies non-players and referees as “person”. The dataset used is designed to enhance detection accuracy, enable the model to autonomously disregard non-players outside the court boundaries, and differentiate referees from players without the need for additional coding, thereby improving the model's performance in live sports analytics.

In light of the challenges identified with our current model, the "Football Player Detection Image Dataset" from Roboflow serves as an instrumental resource for advancing our project. This dataset not only facilitates the improved detection and differentiation of players, referees, and non-participants but also streamlines the data preparation process by offering the ability to download data in various formats, ready for integration with different models. With 612 annotated images capturing detailed aspects such as player positions, referee roles, and ball dynamics, the dataset, though small, provides a solid foundation for initiating a training trial with the YOLO model. This trial aims to refine the model's capabilities to produce accurate and reliable outputs essential for our sports analytics objectives.

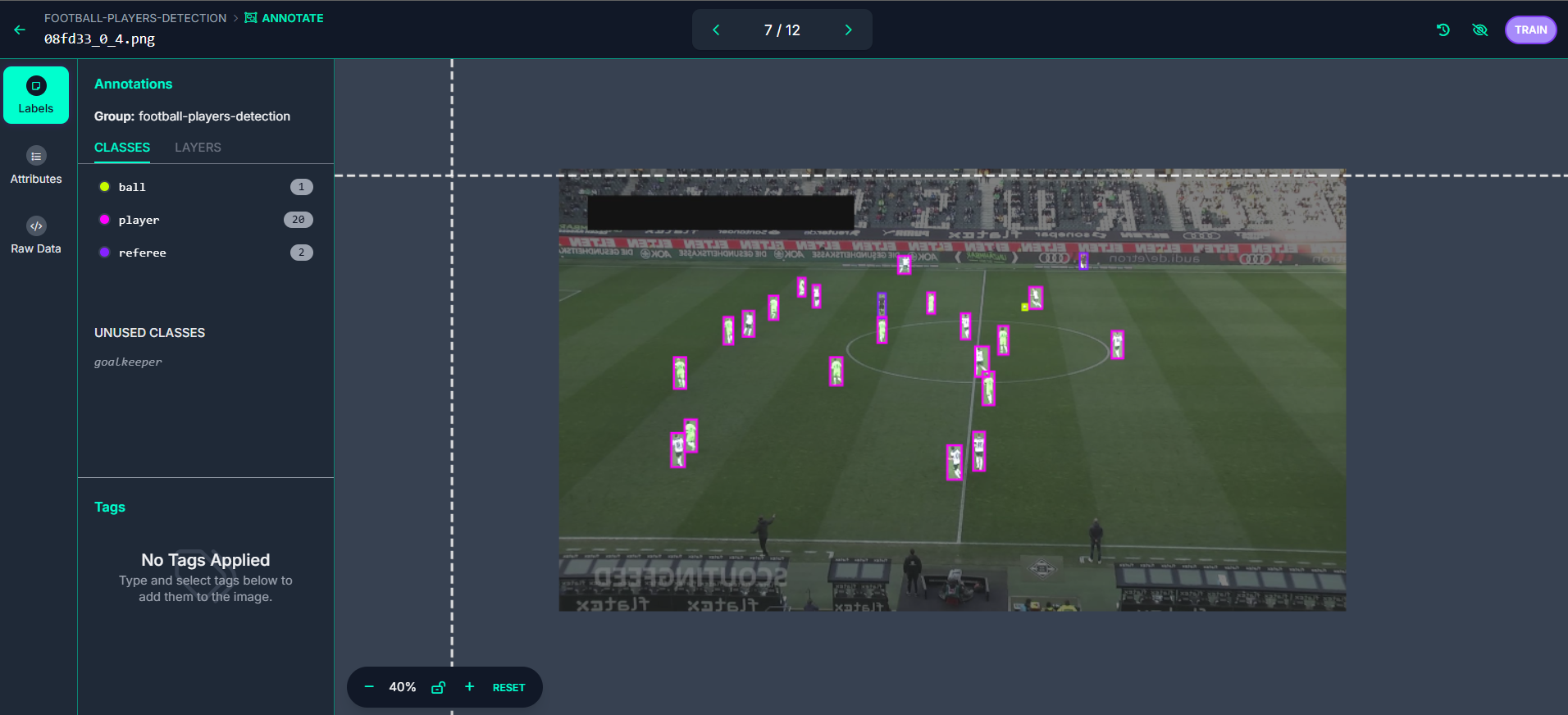


Figure 2: Sample Annotation from the Football Player Detection Image Dataset

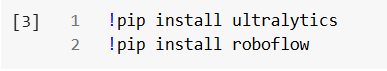
This screenshot from the dataset accessed via Roboflow demonstrates a football game scenario, where players are identified with bounding boxes and non-participants outside the court are not detected. The image also distinctly annotates referees and sideline referees, as well as the football and the goalkeeper, who is specifically marked not just as a player but as a goalkeeper. This level of detail in the annotations underscores the dataset’s capability for precise recognition and differentiation of various roles on the field, enhancing the model's accuracy for sports analytics.

### 2.3.1- downloading the training dataset:

To set up the training environment for the football detection model using YOLO V5, a new folder named "training" will be created, and a Jupyter notebook file, football\_training\_YOLOv5.ipynb, will be initiated. YOLO V5 has been chosen due to its superior accuracy in object detection, particularly for tracking the football.

The setup involves installing necessary libraries such as UltraLytics and Roboflow, which facilitates the dataset download. Once the environment is prepared, the dataset can be accessed via Roboflow by following a simple download process, ensured by the usage of an API key provided on the platform.

After integrating the dataset into the project, it becomes available in the "football\_player\_detections\_1" folder, positioning it for immediate use in model training. This approach aims to optimize the detection capabilities of the YOLO V5 model, focusing on accuracy and reliability in recognizing football-related activities.





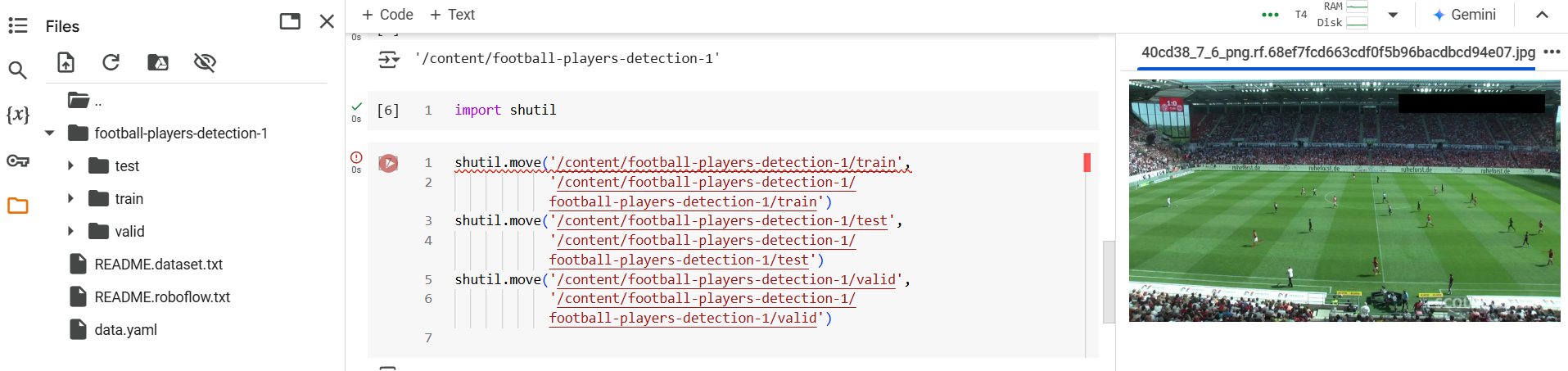
Once the dataset is successfully integrated into the project environment, it will be organized into training, testing, and validation sets. Opening the validation set provides a closer look at the images, which mirror the type of footage anticipated for model application, maintaining consistency in camera angles and scene composition.

The dataset includes labels associated with each image, which are crucial for training the detection model. Each label starts with a class ID indicating the category - such as a player, referee, or ball - followed by coordinates for a bounding box. This bounding box is specified by the center coordinates (X, Y), width, and height, all relative to the image frame dimensions. For example, a width value of 0.32 suggests the object occupies about 32% of the frame's width.

Additionally, the data.yaml file is integral to the setup, as it details the class names and the paths to the image files. This configuration ensures that the model is accurately informed about where to find and how to interpret the dataset. By understanding the dataset's structure and label schema, you can efficiently train and validate the model to achieve high accuracy in detecting and differentiating between various subjects in football game footage.

To configure the training environment for our football detection model using YOLO V5, we must structure our dataset into a specific folder hierarchy expected by the training code. This precise configuration involves placing the "football\_player\_detections\_1" directory within an identically named folder, ensuring the path is correctly nested. This folder structure is critical as the training script depends on this setup to function properly.

To automate this reorganization and maintain consistency across different setups, we employed Python's shutil library to programmatically adjust the directory structure. This method not only prevents potential errors during manual setup but also enhances the reproducibility of the project setup, facilitating straightforward scaling and collaboration. Once the dataset is correctly positioned in the "football\_player\_detections\_1/football\_player\_detections\_1" folder, it is primed for immediate integration and use in training the YOLO V5 model, optimizing its capabilities for accurate and reliable detection of football-related activities.



### 2.3.2- Training the model:

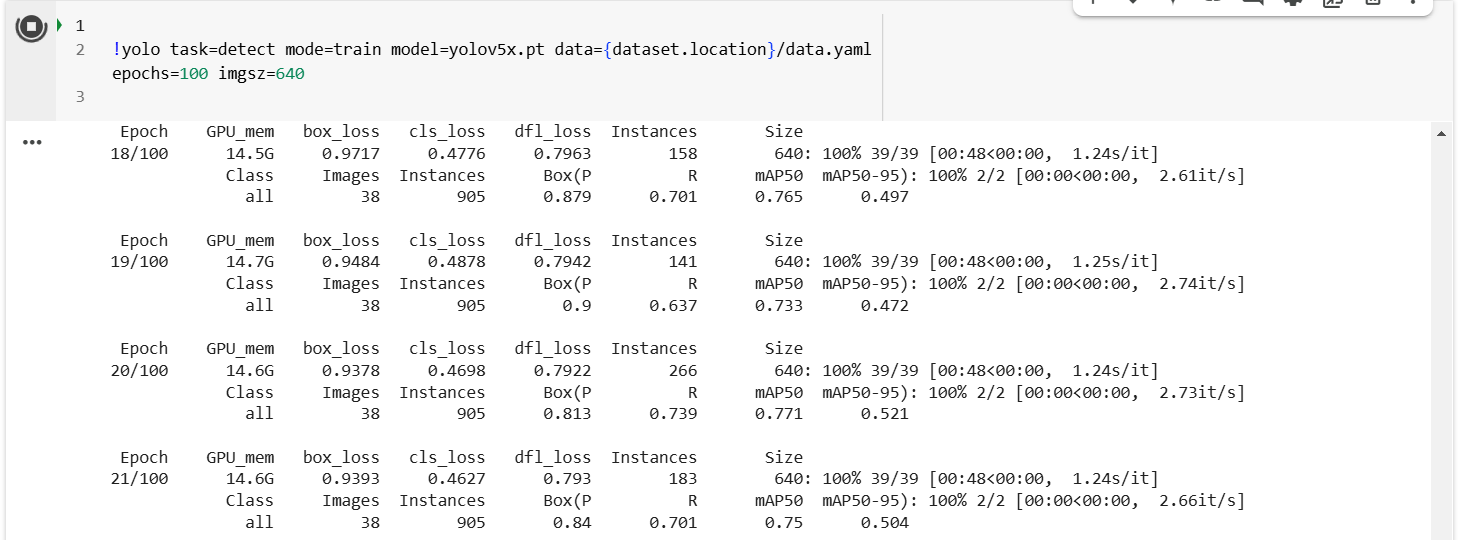
To initiate the training of our football detection model using YOLO V5, a specific sequence of operations is executed. We utilize the Ultralytics command-line interface to set up and run our model training. The command used specifies YOLO as the application, with the task set to detection mode appropriately set to 'train'. For this project, the choice of model variant is YOLO V5 X (extra-large), optimized for higher accuracy at the cost of increased computational demand.

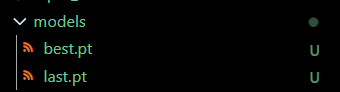
The command further includes the location of our dataset, linked through the 'data.yaml' file, which directs the model to the necessary training, validation, and test sets. We also configure the number of training epochs with the --epochs parameter and set the image resolution to 640 pixels, ideal for our detection tasks.

Given the absence of a local GPU, the training is conducted on Google Colab, which offers free access to a GPU, facilitating a more efficient training process. The setup on Google Colab involves creating a new notebook where the training script and necessary installation commands are input. This environment supports running extensive computations without the need for personal hardware capabilities.

Throughout the training phase, progress is monitored by observing the loss metrics, which ideally should decrease, indicating effective learning. Upon completion, the trained model weights ('best' and 'last') are saved automatically under the 'runs/train' directory. These can be downloaded directly from Colab or transferred to Google Drive for persistent storage.

Finally, these weights are integrated back into the project under a newly created 'models' folder within the local environment. This setup not only encapsulates the training process but also prepares the model for subsequent deployment and testing, ensuring that the model's performance can be evaluated and utilized in practical scenarios.





To evaluate the performance of our fine-tuned YOLO V5 model, we proceed by testing it using a new set of inputs to observe its capability in real-world scenarios. This involves running the model with the 'best.pt' weights file, which contains the parameters that yielded the best validation performance during training. The process is straightforward: adjust the model path in the script to load the 'best.pt' file, save the changes, and execute the run.

Upon running the model, the output provides a visual confirmation of its predictive accuracy. The results demonstrate that the model effectively distinguishes between different entities on the football field—players, referees, and the ball—with high precision. Notably, the model has improved in detecting the ball, especially during dynamic play, where previous models showed deficiencies. The output includes the detection of players as players, referees as referees, and an enhanced ability to track the ball, all marked accurately with bounding boxes.

The output directory, labeled as 'predict2' under 'runs/detect', stores the results of this testing phase. Opening this directory reveals the processed images where the model's predictions are visualized. These images confirm that non-players outside the court remain undetected, maintaining focus on relevant game participants. This testing not only validates the model's effectiveness post-training but also showcases its readiness to be deployed in environments requiring accurate real-time sports analysis.

