Group 15's (AASD 4016)

Full Stack Data Science Systems Project

Soccer Offside Detection Implementation

<u>Team members:</u>

1. Kareem Albeetar - 101448256
2. Chun Lam Cheung - 101471424
3. Eltigani Hamadelniel - 101440790
4. Yi Liu - 101493612

Problem Background

Despite advancements in sports technology, accurately detecting soccer objectives, particularly offside situations, remains a persistent challenge. Current detection systems often suffer from inaccuracies, leading to contentious calls that affect game outcomes and spectator experience. As a result, there is a critical need for an innovative solution that improves the precision and reliability of offside detection in soccer matches, enhancing fairness, integrity, and the overall quality of the game.

Significance

1. **Fairness & Integrity** - Implementing an accurate soccer objective detection system is paramount for upholding the integrity of the game.
2. **Loss of Fan Engagement & Viewership** - inaccurate offside calls can lead to frustration among fans and diminish their engagement with the sport.
3. **Revenue & Sponsorships -** Reduced fan engagement as a result of inaccurate offside calls can translate into a loss of revenue due to dips in pay-per-view sales and match ticket sales.

Present options

FIFA's Semi-Automated Offside Detector

The only present solution for automating detecting offside plays uses cameras to track the ball and players 50 times per second, calculating their positions. A sensor inside the ball transmits data to detect the exact kickpoint. With this collected data the system generates automated offside alerts for video officials to review. Officials validate the alerts before informing the referee.

Assistant Referees

Assistant Referees, commonly referred to as linesmen in soccer games, are officials positioning along the sidelines, usually one on each side of the field, assistant referees closely monitor play to make decisions regarding offside infractions, throw-ins, and fouls occurring near their respective touchlines.

Methodology

Ball & Player Detection - We used ultralytics to train two YOLOv8 models. One trained with annotated image dataset that include classes of players, goalkeepers, referees and the ball. The second expanded the dataset and focused on detecting the ball and was trained with annotated images that only include the ball and players. Using the output detection boxers we created annotator classes to draw markers for each class detected within the input image.

Agglomerative Clustering - In order to differentiate between the detected players we employed the use of an agglomerative clustering model that is fed the colour histograms of cropped images taken from the center of each bounding box. The algorithm then returns labels classifying each box based on their respective pixel intensities. The resultant team labels are assigned to the detection objects as attributes and annotated within the image.

Vanishing Point Calculation **-** The Vanishing Point method is to calculate the intersection point for 2 lines that was captured by the generated cv2.houghline. Before the process, the original image will turn the image into HSV, mask the green part of the image as a field and build edges with cv2.CannyEdge. After that, the edge image is ready to feed.

Offside Calculate **-** Calculation of the angle between a vanishing point and a test point in an image. It considers a reference point and a desired direction (left or right). It does this by converting points to vectors, finding the angle between them, and adjusting for the reference point and desired direction. In the defending team, the minimum angle is the last man, and for the attacking team the angle value less than last man, then will be labeled offside. When the coordinate of the offside player is matched with the player in possession, the flag will rise.

Deployment

For our deployment the following technology stack was used:

- **Dataset** - Augmentation was applied on the annotated dataset using Roboflow's workspace platform
- **Model** - We leveraged Ultralytics to fine-tuned a pre-trained YOLOv8 model
- **Training Environment** - The model was trained on a host colab runtime leveraging a T4
- **Flask** - We leveraged Flask to create our web application scripting our code with an Object Oriented Methodology
- **Docker** - We created a docker image and pushed that image into our Docker Hub repository.
- **Virtual Machine** - We created a virtual machine instance using GCP which hosted our docker.