1. Introduction

The task of player image segregation involves the classification of multiple player images into distinct groups based on visual similarity. This project utilizes object detection and feature extraction techniques, followed by clustering to categorize the images into four predefined classes. While clustering-based approaches were successfully implemented, various alternative approaches, including cosine similarity and fine-tuning a pre-trained model, were considered but ultimately not implemented due to time constraints and limitations in available data.

2. Implemented Approach

2.1 Object Detection Using YOLOv8

The implemented approach begins with detecting and segmenting players in a reference image using the YOLOv8 segmentation model (yolov8s-seg.pt). The YOLO model accurately identifies the players, returning bounding boxes for each detected player. Using these bounding boxes, each player is cropped and saved as individual images (player0.png, player1.png, player2.png, and player3.png), which are later used as reference images.

2.2 Feature Extraction Using Pre-Trained Model (MobileNet-V2)

For the extracted images, a pre-trained MobileNet-V2 model is used to extract feature vectors, which provide a high-dimensional representation of each image. This model, loaded through TensorFlow Hub, is known for its efficiency in feature extraction from images, making it suitable for tasks requiring visual similarity comparisons. The images are resized, normalized, and passed through the model to generate feature vectors.

2.3 Clustering Using K-Means

Once the feature vectors for both the reference images and target images (from two folders: two_players_bot and two_players_top) are extracted, the K-Means clustering algorithm is applied to group the images into four distinct clusters. K-Means, an unsupervised learning algorithm, partitions the images based on their feature vectors, ensuring that visually similar images are grouped together.

2.4 Folder Segregation

After clustering, an output folder is created, containing four subfolders corresponding to each cluster (i.e., player0, player1, player2, player3). Images from the two input folders are then moved into their respective subfolders based on the clusters they are assigned to by the K-Means algorithm.

---

3. Alternative Approaches Considered

3.1 Cosine Similarity Approach

One alternative approach initially considered was to use **cosine similarity** to compare the feature vectors of the images. Cosine similarity measures the cosine of the angle between two vectors and provides a similarity score, which ranges between -1 (completely dissimilar) to 1 (completely similar). The idea behind this approach was to compute the cosine similarity between each target image and the reference

images (player0.png, player1.png, player2.png, player3.png) and assign each target image to the reference player with the highest similarity score.

Cosine Similarity function:

```
def cosineSim(a1,a2):
    sum = 0
    suma1 = 0
    sumb1 = 0
    for i,j in zip(a1, a2):
        suma1 += i * i
        sumb1 += j*j
        sum += i*j
    cosine_sim = sum / ((sqrt(suma1))*(sqrt(sumb1)))
    return cosine_sim
```

3.2 Fine-Tuning a Pre-Trained Model

Another proposed approach was to fine-tune the pre-trained model “dinov2” using a small set of labeled player images. Fine-tuning involves updating the weights of a pre-trained model with domain-specific data, without updating all the weights. Only few layers of weights will be unfreeze and updated on the new data to make it task specific. Fine-tuning could potentially improve the accuracy of the extracted feature vectors by better adapting the model to the task of player image classification. The only reason, it was not used is due to low data labels present.

---

4. Additional Approaches Considered

4.1 Hierarchical Clustering

Hierarchical clustering is another clustering technique that could have been considered. Unlike K-Means, which requires specifying the number of clusters beforehand, hierarchical clustering creates a tree-like structure of clusters, allowing one to visualize the relationships between different images.

Advantages:

- **No Predefined Cluster Number**: This method doesn't require specifying the number of clusters at the start. One could explore different cluster groupings by cutting the hierarchical tree at different levels.
- **Visual Representation**: The dendrogram produced by hierarchical clustering offers an intuitive way to visualize how images are grouped at various levels of similarity