**PROJECT: Tennis Ball Detection and Tracking Application**

In this project, I used **Roboflow** to annotate data for training a YOLOv5 model to detect tennis balls.

1. **Convert Videos to Frames**: I uploaded my tennis videos to Roboflow. Roboflow automatically provides the option to extract frames from the uploaded videos. Look for the **"Frames"** option in the dataset interface.Set the desired frame extraction rate .Start the frame extraction process. Roboflow will convert videos into a series of image frames, which can be viewed and managed in the project.
2. **Annotate Data**: Using Roboflow’s annotation tools, I marked the positions of tennis balls in each frame. This involved drawing bounding boxes around the balls and labeling them accordingly.
3. **Export Dataset**:Export the annotated dataset in YOLOV5 format for use in Google Colab.
4. **Clean and Normalize the Data**: After annotating, Roboflow automatically handles data cleaning and normalization during the export process. This means that the images are prepared and optimized for training without requiring additional steps.
5. **Set Up YOLOv5 in Google Colab:**

* **Clone YOLOv5 Repository:**Open Google Colab and run the following commands to clone the YOLOv5 repository:

!git clone https://github.com/ultralytics/yolov5.git

%cd yolov5

!pip install -r requirements.txt

6**: Configure YOLOv5 Architecture**

***# define number of classes based on YAML***

import yaml

with open(dataset.location + "/data.yaml", 'r') as stream:

num\_classes = str(yaml.safe\_load(stream)['nc'])

***#customize iPython writefile so we can write variables***

from IPython.core.magic import register\_line\_cell\_magic

@register\_line\_cell\_magic

def writetemplate(line, cell):

with open(line, 'w') as f:

f.write(cell.format(\*\*globals()))

**Create a Custom Configuration File**:

# Ultralytics YOLOv5 🚀, AGPL-3.0 license

%%writetemplate /content/yolov5/models/custom\_yolov5s.yaml

# Parameters

nc: {num\_classes} # number of classes

depth\_multiple: 0.33 # model depth multiple

width\_multiple: 0.50 # layer channel multiple

anchors:

  - [10, 13, 16, 30, 33, 23] # P3/8

  - [30, 61, 62, 45, 59, 119] # P4/16

  - [116, 90, 156, 198, 373, 326] # P5/32

# YOLOv5 v6.0 backbone

backbone:

 # [from, number, module, args]

  [[-1, 1, Conv, [64, 6, 2, 2]], # 0-P1/2

    [-1, 1, Conv, [128, 3, 2]], # 1-P2/4

    [-1, 3, C3, [128]],

    [-1, 1, Conv, [256, 3, 2]], # 3-P3/8

    [-1, 6, C3, [256]],

    [-1, 1, Conv, [512, 3, 2]], # 5-P4/16

    [-1, 9, C3, [512]],

    [-1, 1, Conv, [1024, 3, 2]], # 7-P5/32

    [-1, 3, C3, [1024]],

    [-1, 1, SPPF, [1024, 5]], # 9 ]

# YOLOv5 v6.0 head

head: [

    [-1, 1, Conv, [512, 1, 1]],

    [-1, 1, nn.Upsample, [None, 2, "nearest"]],

    [[-1, 6], 1, Concat, [1]], # cat backbone P4

    [-1, 3, C3, [512, False]], # 13

[-1, 1, Conv, [256, 1, 1]],

    [-1, 1, nn.Upsample, [None, 2, "nearest"]],

    [[-1, 4], 1, Concat, [1]], # cat backbone P3

    [-1, 3, C3, [256, False]], # 17 (P3/8-small)

[-1, 1, Conv, [256, 3, 2]],

    [[-1, 14], 1, Concat, [1]], # cat head P4

    [-1, 3, C3, [512, False]], # 20 (P4/16-medium)

[-1, 1, Conv, [512, 3, 2]],

    [[-1, 10], 1, Concat, [1]], # cat head P5

    [-1, 3, C3, [1024, False]], # 23 (P5/32-large)

[[17, 20, 23], 1, Detect, [nc, anchors]], # Detect(P3, P4, P5)]