### IMPORTING DEPENDENCIES AND DATASETS

```
import torch
```

```
!git clone https://github.com/Susa-43/Sports_Classification_dataset.git
```

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

### UNZIPPING THE DATASET

```
!unzip Sports_Classification_dataset/data.zip
```

### INSTALLING REQUIREMENTS FOR THE USAGE OF YOLOV5

```
%cd yolov5
```

```
!pip install -r requirements.txt
```

### TRAINING THE YOLOV5 MODEL FOR SPORTS

```
!python classify/train.py --model yolov5s-cls.pt --data ../data --epochs 20 --img 224 --batch 15
```

### PLOTTING ACCURACY AND LOSS GRAPH

```
import pandas as pd

# read the results.csv file
df = pd.read_csv('runs/train-cls/exp/results.csv')

# print the column names
print(df.columns)
df
```

```
import pandas as pd
import matplotlib.pyplot as plt

# read the results.csv file
df = pd.read_csv('runs/train-cls/exp/results.csv')

# extract the accuracy and loss values from the dataframe
train_acc = df['  metrics/accuracy_top1']
train_loss = df['             train/loss']
val_loss = df['               val/loss']

# plot the accuracy graph
plt.plot(train_acc, label='Train')
plt.title('Classification Accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.show()

# plot the loss graph
plt.plot(train_loss, label='Train')
plt.plot(val_loss, label='Validation')
plt.title('Classification Loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()
plt.show()
```

### PREDICTING THE SPORTS USING THE TRAINED MODEL

```
!python classify/predict.py --weights runs/train-cls/exp/weights/best.pt --source ../Sports_Classification_dataset/cricket.mp4
```

### POSE ESTIMATION, FEATURE EXTRACTION FOR THE SPORTS VIDEO AND CONVERING THE OUTPUT INTO VIDEO

```
!pip install mediapipe
```

```
%cd ../


import cv2
import mediapipe as mp
import numpy as np

mp_drawing= mp.solutions.drawing_utils
mp_pose= mp.solutions.pose

def calculate_angle(a,b,c):
    a = np.array(a) # First
    b = np.array(b) # Mid
    c = np.array(c) # Last

    radians = np.arctan2(c[1]-b[1], c[0]-b[0]) - np.arctan2(a[1]-b[1], a[0]-b[0])
    angle = np.abs(radians*180.0/np.pi)

    if angle >180.0:
        angle = 360-angle

    return angle

cap = cv2.VideoCapture(r'yolov5/runs/predict-cls/exp/cricket.mp4')
filename="final.avi"
codec=cv2.VideoWriter_fourcc('X','V','I','D')
width  = int(cap.get(3))
height = int(cap.get(4))
fps=24
resolution=(width,height)
out_video=cv2.VideoWriter(filename,codec,fps,resolution)# To convert the frames to video

with mp_pose.Pose(min_detection_confidence=0.5,min_tracking_confidence=0.5) as pose :
    while cap.isOpened():
        ret,image=cap.read()
        if ret == False:
          break

        results=pose.process(image)

        try:
            landmarks = results.pose_landmarks.landmark

            #Get coordinates
            #for elbow

            #for left
            shoulderleft = [landmarks[mp_pose.PoseLandmark.LEFT_SHOULDER.value].x,landmarks[mp_pose.PoseLandmark.LEFT_SHOULDER.value].y]
            elbowleft = [landmarks[mp_pose.PoseLandmark.LEFT_ELBOW.value].x,landmarks[mp_pose.PoseLandmark.LEFT_ELBOW.value].y]
            wristleft = [landmarks[mp_pose.PoseLandmark.LEFT_WRIST.value].x,landmarks[mp_pose.PoseLandmark.LEFT_WRIST.value].y]
            #for right
            shoulderright = [landmarks[mp_pose.PoseLandmark.RIGHT_SHOULDER.value].x,landmarks[mp_pose.PoseLandmark.RIGHT_SHOULDER.value].
            elbowright = [landmarks[mp_pose.PoseLandmark.RIGHT_ELBOW.value].x,landmarks[mp_pose.PoseLandmark.RIGHT_ELBOW.value].y]
            wristright = [landmarks[mp_pose.PoseLandmark.RIGHT_WRIST.value].x,landmarks[mp_pose.PoseLandmark.RIGHT_WRIST.value].y]

            #for knee

            #for left
            hipleft = [landmarks[mp_pose.PoseLandmark.LEFT_HIP.value].x,landmarks[mp_pose.PoseLandmark.LEFT_HIP.value].y]
            kneeleft = [landmarks[mp_pose.PoseLandmark.LEFT_KNEE.value].x,landmarks[mp_pose.PoseLandmark.LEFT_KNEE.value].y]
            ankleleft = [landmarks[mp_pose.PoseLandmark.LEFT_ANKLE.value].x,landmarks[mp_pose.PoseLandmark.LEFT_ANKLE.value].y]
            #for right
            hipright = [landmarks[mp_pose.PoseLandmark.RIGHT_HIP.value].x,landmarks[mp_pose.PoseLandmark.RIGHT_HIP.value].y]
            kneeright = [landmarks[mp_pose.PoseLandmark.RIGHT_KNEE.value].x,landmarks[mp_pose.PoseLandmark.RIGHT_KNEE.value].y]
            ankleright = [landmarks[mp_pose.PoseLandmark.RIGHT_ANKLE.value].x,landmarks[mp_pose.PoseLandmark.RIGHT_ANKLE.value].y]


            # Calculate angle
            angle1 = int(calculate_angle(shoulderleft, elbowleft, wristleft))
            angle2 = int(calculate_angle(shoulderright, elbowright, wristright))
            angle3 = int(calculate_angle(hipleft, kneeleft, ankleleft))
            angle4 = int(calculate_angle(hipright, kneeright, ankleright))

            # Print angle in the video
            cv2.putText(image, str(angle1),
                           tuple(np.multiply(elbowleft, [width, height]).astype(int)),
                           cv2.FONT_HERSHEY_SIMPLEX, 3, (255, 255, 255), 3, cv2.LINE_AA
                                )
            cv2.putText(image, str(angle2),
                           tuple(np.multiply(elbowright, [width, height]).astype(int)),
                           cv2.FONT_HERSHEY_SIMPLEX, 3, (255, 255, 255), 3, cv2.LINE_AA
```

```
                            )
            cv2.putText(image, str(angle3),
                        tuple(np.multiply(kneeleft, [width, height]).astype(int)),
                        cv2.FONT_HERSHEY_SIMPLEX, 3, (255, 255, 255), 3, cv2.LINE_AA
                            )
            cv2.putText(image, str(angle4),
                        tuple(np.multiply(kneeright, [width, height]).astype(int)),
                        cv2.FONT_HERSHEY_SIMPLEX, 3, (255, 255, 255), 3, cv2.LINE_AA
                            )
        except:
            pass

        mp_drawing.draw_landmarks(image,results.pose_landmarks,mp_pose.POSE_CONNECTIONS,mp_drawing.DrawingSpec(color=(0,0,255),thickness=
        out_video.write(image)

cv2.destroyAllWindows()
out_video.release()
cap.release()
```