



# 深度學習智慧應用

## Lab 3: YOLO-Pose

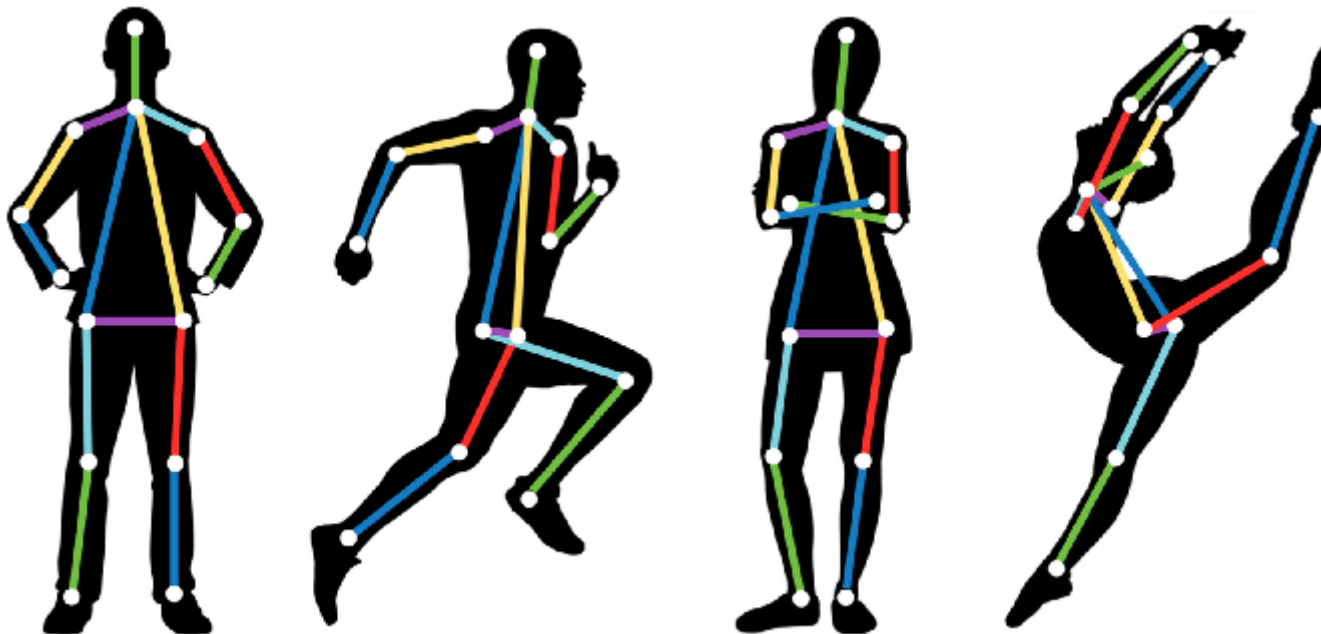
Network Optimization Lab (NOL)

Department of Computer Science

National Chiao Tung University

# What is Human Pose Estimation?

- Human pose estimation is a computer vision task that locates and tracks **keypoints** on the human body to estimate its pose or configuration.

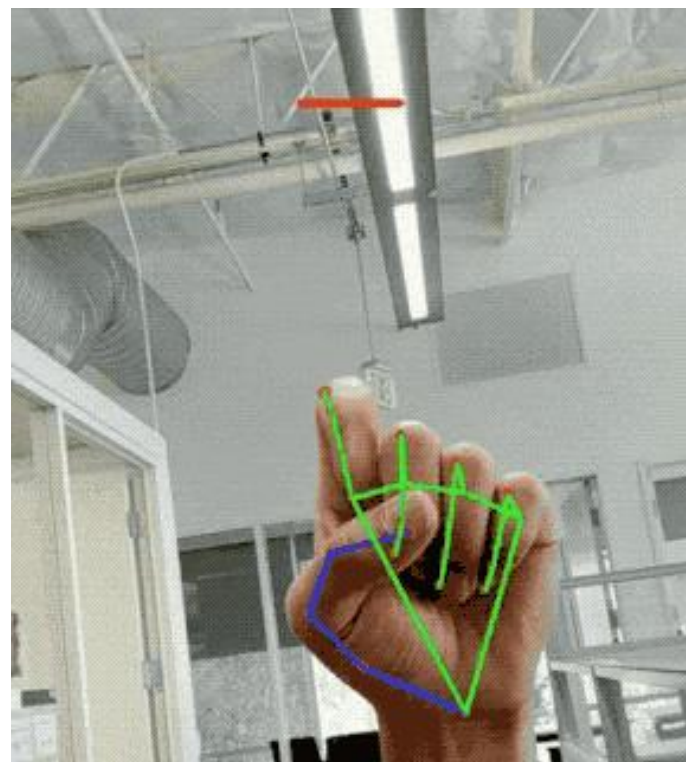


Ref: <https://www.analyticsvidhya.com/blog/2021/10/human-pose-estimation-using-machine-learning-in-python/>

# Applications of Human Pose Estimation

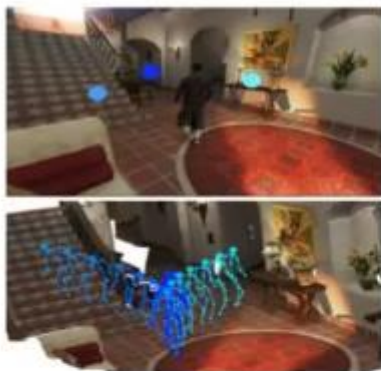


Ref: <https://viso.ai/deep-learning/pose-estimation-ultimate-overview/>



Ref: <https://technews.tw/2019/08/21/google-on-device-real-time-hand-tracking-with-mediapipe/>

# Applications of Human Pose Estimation



Action prediction



Surveillance



Cloth Parsing



Online Coaching



Movie and Game



AR and VR



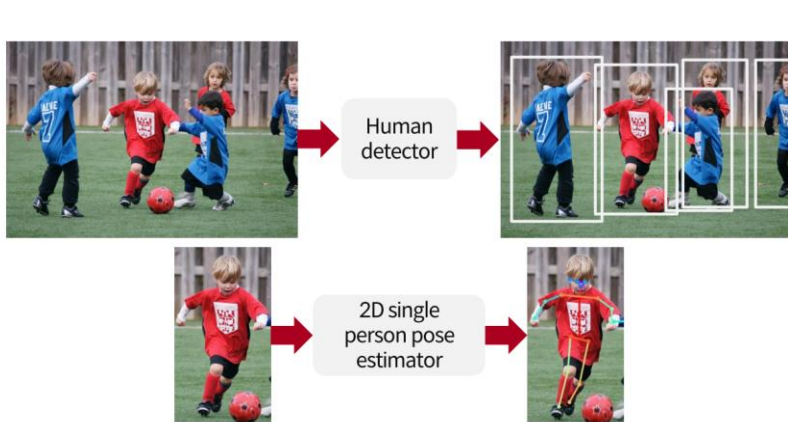
Healthcare

Ref: <https://viso.ai/deep-learning/pose-estimation-ultimate-overview/>

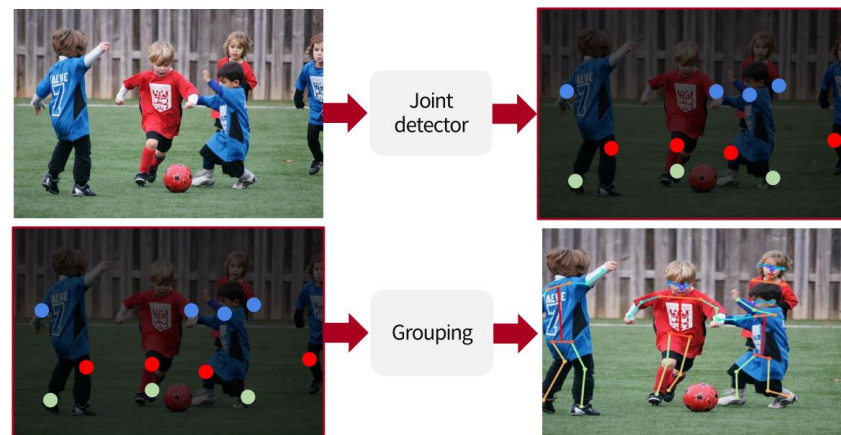


# Human Pose Estimation Methods

	Top-down Approach	Bottom-up Approach
Core Idea	Detect people first, then estimate poses	Detect all keypoints first, then group them into people
Computational Efficiency	Lower for fewer people, higher cost for more people	Relatively fixed, regardless of the number of people



Top-down



Bottom-up

Ref: <https://velog.io/@dangdang2222/Top-down-approach-vs-Bottom-up-approach>

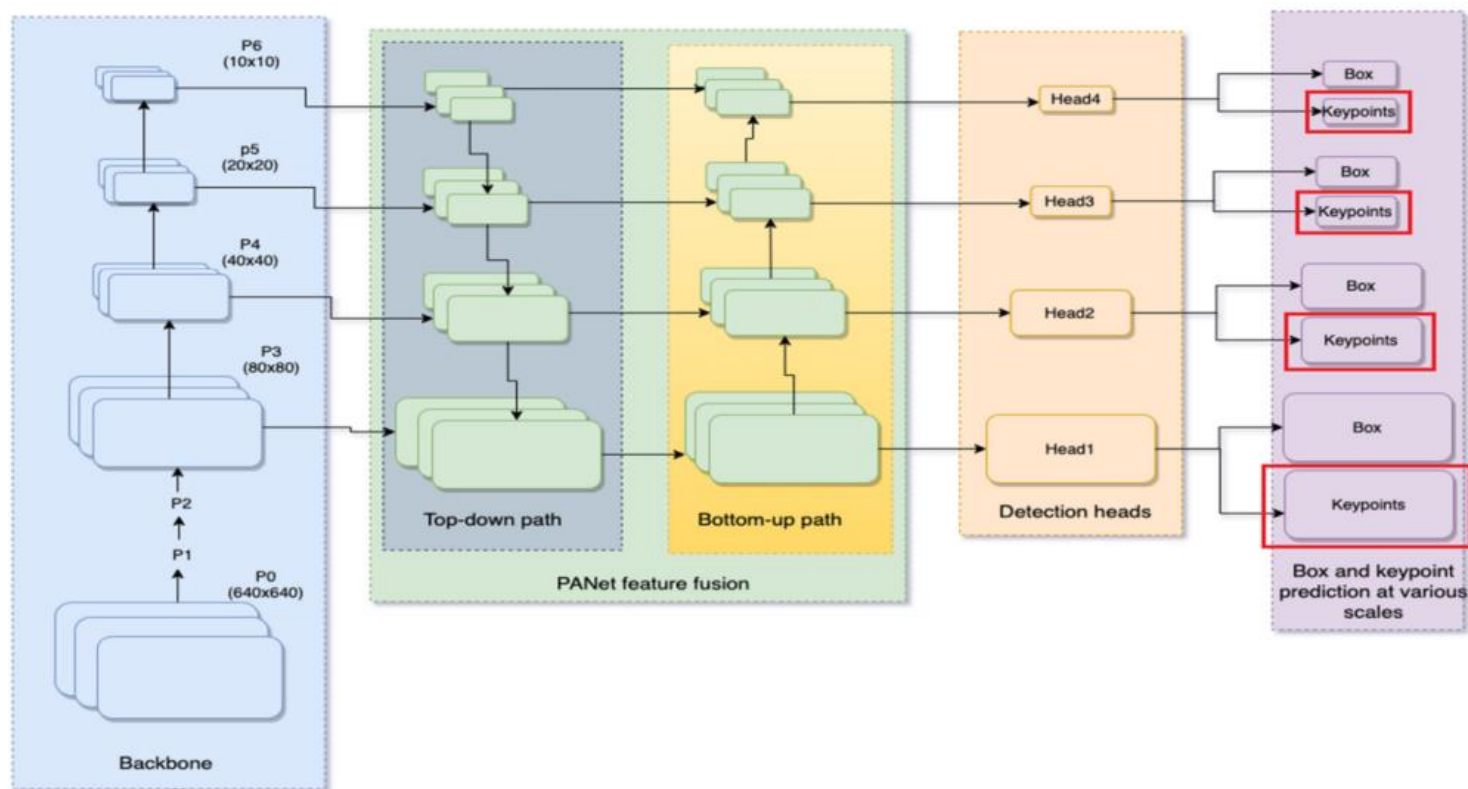


# Human Pose Estimation Methods

	Top-down Approach	Bottom-up Approach
Core Idea	Detect people first, then estimate poses	Detect all keypoints first, then group them into people
Computational Efficiency	Lower for fewer people, higher cost for more people	Relatively fixed, regardless of the number of people
Feature	- High accuracy	- High efficiency
Example	MediaPipe, AlphaPose	OpenPose, DensePose

# YOLO-Pose

- The Architecture of YOLO-Pose



Maji, Debapriya, et al. "Yolo-pose: Enhancing yolo for multi person pose estimation using object keypoint similarity loss." *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022.

# Understanding YOLO-Pose

- **Approach:**

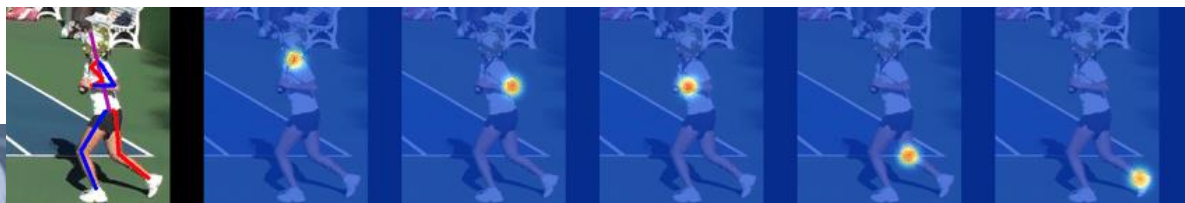
Bottom-up style, directly predicts skeleton keypoints.

- **Difference from Traditional Bottom-up:**

- No heatmap generation.
- Keypoints are directly associated with predicted bounding boxes (anchors).
- Eliminates the need for grouping steps.

- **Workflow**

1. Predicts bounding boxes and associated keypoints in one step.
2. Skeletons are automatically grouped with the bounding box as a natural unit.





# Outputs

Taking the default YOLO-Pose model as an example

In skeleton detection, **17 coordinate points** are generated, which represent the following in order:

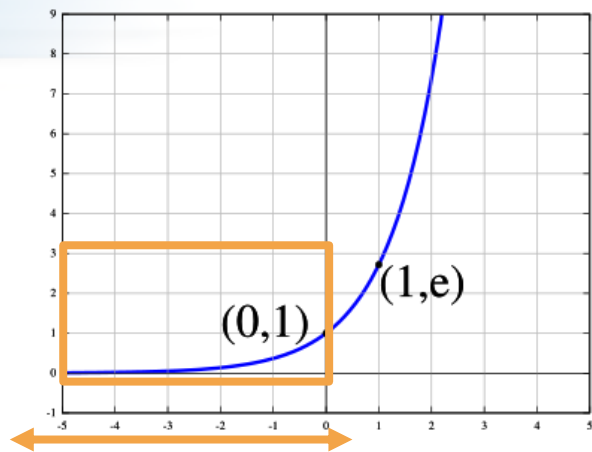


ID	body parts	ID	body parts
0	Nose	9	Left wrist
1	Left eye	10	Right wrist
2	Right eye	11	Left hip
3	Left ear	12	Right hip
4	Right ear	13	Left knee
5	Left shoulder	14	Right knee
6	Right shoulder	15	Left ankle
7	Left elbow	16	Right ankle
8	Right elbow		

# Evaluation

## • OKS (Object Keypoint Similarity)

$$OKS = \frac{\sum_{i=1}^N \exp\left(-\frac{d_i^2}{2s^2k_i^2}\right) \delta(v_i > 0)}{\sum_{i=1}^N \delta(v_i > 0)}$$



$d_i$  愈大負值越大，  
指數函數結果愈接  
近0，預測結果差

$d_i$  愈小負值越小，  
指數函數結果愈接  
近1，預測結果好

- $N$ : 總關鍵點數量
- $d_i$ : 第  $i$  個關鍵點的歐幾里得距離(預測與真值的差距)  $d_i = \sqrt{(x_{pred} - x_{gt})^2 + (y_{pred} - y_{gt})^2}$
- $s$ : 標記物體的比例(如邊界框面積)
- $k_i$ : 關鍵點的尺度正規化參數，預設值根據每個關鍵點的重要性給定(例如，頭部、膝蓋等的不同敏感度)
- $v_i$ : 關鍵點的可見性標記(0: 未標記(可能在圖片外)，1: 標記但被遮擋，2: 標記且可見)
- $\delta(v_i > 0)$ : 只對可見或遮擋的點進行計算。



# Lab 3



# Lab3

- Data Label
- Training
- Performance
- Demo

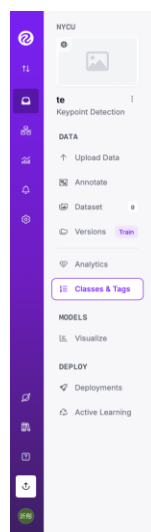


# Data Label

- Labeling tool
  - Roboflow => <https://app.roboflow.com/>
- Unlabeled Dataset
  - Google Drive => [https://drive.google.com/drive/folders/1JEw733qWBU-fEu0m\\_97i\\_uJyw\\_lk5YIL?usp=share\\_link](https://drive.google.com/drive/folders/1JEw733qWBU-fEu0m_97i_uJyw_lk5YIL?usp=share_link)
  - Please note that each group has its own unlabeled training videos in the group file.
  - The images in "Fore\_Back\_Detection" file are shared among groups for forehand and backhand detection.

# Steps Different from Lab2

- Choose "Keypoints Detection".



## Classes & Tags

Add the class for an object you are detecting. You will be able to add keypoints to your class after creation.

person (the object containing your keypoints)

Upload Classes CSV

Add Class

roboflow

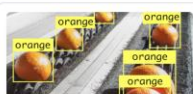
Let's create your project.

NYCU > New Public Project

Project Name Name cannot be empty. License CC BY 4.0

Annotation Group E.g., 'humans' or 'cars' or 'animals'.

Project Type

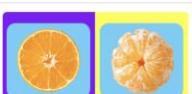


Keypoints Detection

Identify keypoints ("skeletons") on subjects.

Best For

- Pose Estimation



Instance Segmentation

Detect multiple objects and their actual shape.

Best For

- Measurements
- Odd Shapes





Image Classification

Classify the entire image.

Best For

- Measurements
- Odd Shapes



Keypoints Detection

Identify keypoints ("skeletons") on subjects.

Best For

- Pose Estimation

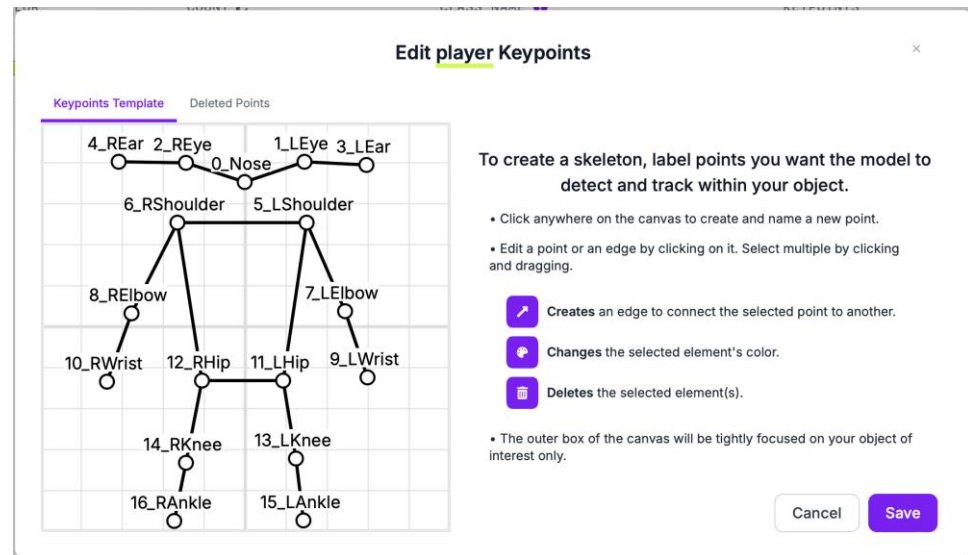
Show More

Cancel Create Public Project

- Define "Classes & Tags".

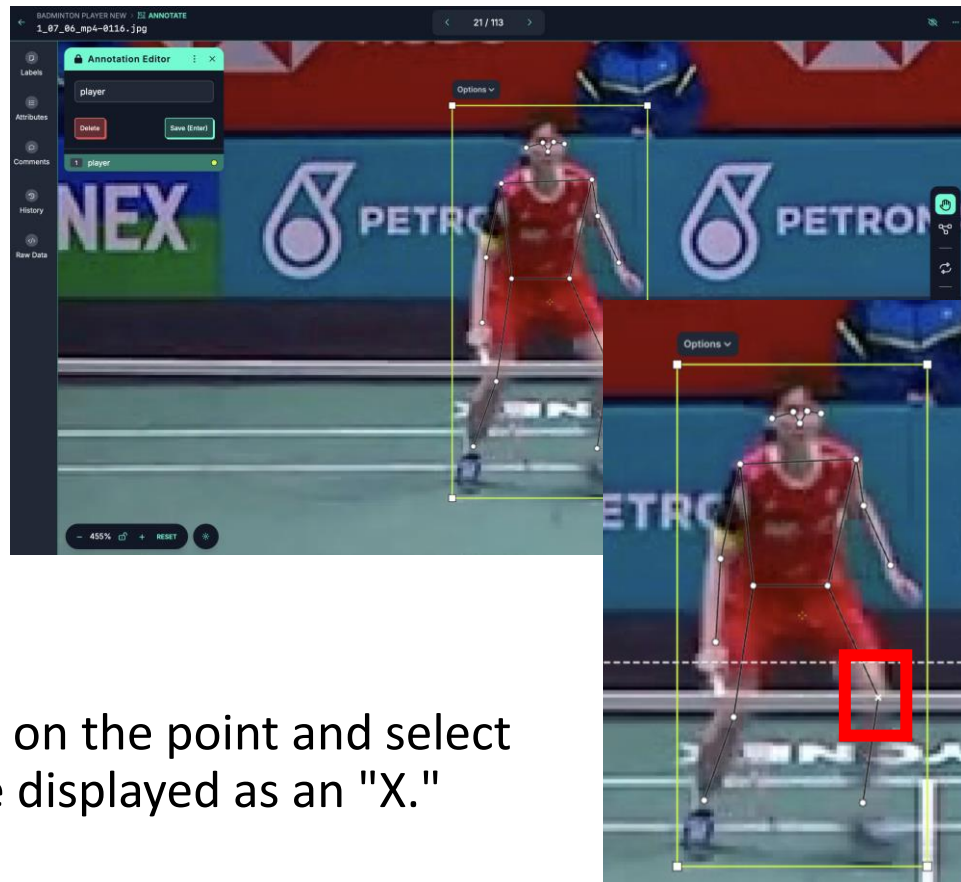
# Steps Different from Lab2(cont)

- There are 17 key points in the graph on the right.
- These points should be created in order, and their names can be assigned as preferred.



# Steps Different from Lab2(cont)

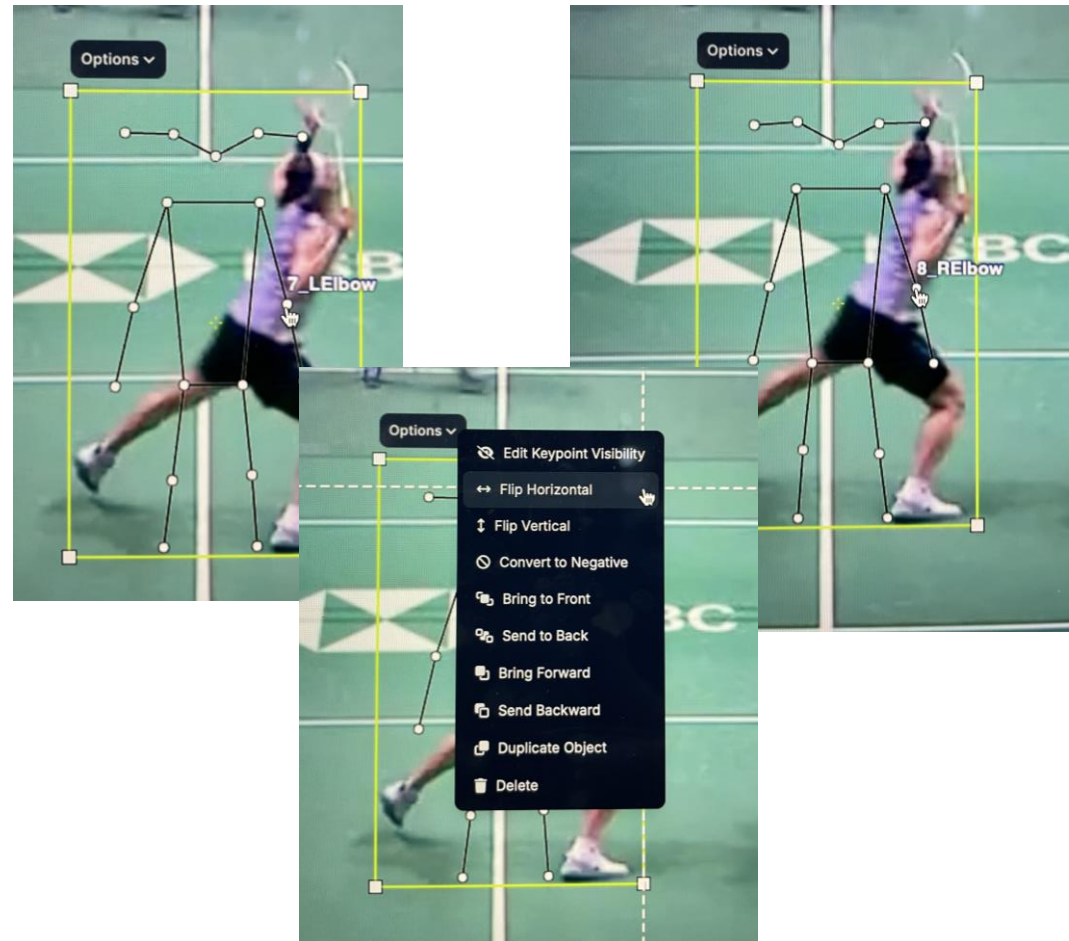
- On the annotation page, once the player is bounded within the screen, the previously defined skeleton will appear inside the bounding box.
- The key points need to be adjusted to align with the corresponding skeleton parts.
- If a key point is occluded, click on the point and select "Mark." The point will then be displayed as an "X."





# Steps Different from Lab2(cont)

- Since the key points differ between the left and right sides, ensure that your labels reflect the correct left and right orientation.
- If needed, you can use the “Options” → “Flip Horizontal” function.



# Trainning and inference

- Ultralytics

- **Environment dependency is the same as in lab2.**

- Train

```
from ultralytics import YOLO
# Load a model
model = YOLO("yolov8n-pose.pt")
# load a pretrained model (recommended for training) # Train the model
results = model.train(data="your_labeling_result.yaml", ...)
```

- Predict

# The output format

- Save the detection results as a JSON file following this format:
  - Filename: results.json
  - Content:

```
{
  "image1.jpg": [
    {
      "bbox": [100.0, 200.0, 150.0, 300.0],
      "keypoints": [
        [120.0, 210.0],
        [125.0, 215.0],
        ...
      ]
    },
    {
      "bbox": [50.0, 180.0, 90.0, 250.0],
      "keypoints": [
        [60.0, 190.0],
        [65.0, 195.0],
        ...
      ]
    }
  ],
  "image2.jpg": [
    ...
  ]
}
```

# Hint: How to save the result

```
# YOLO 允許直接指定一個目錄，直接對目錄下的所有影像進行預測
results = model.predict(source="./Fore_Back_Detection")
results_data = {}
for result in results:
    # 取出預測結果中儲存的影像的路徑，並僅取當中的檔名存為 image_name
    image_name = result.path.split('/')[-1]
    image_data = []

    # 逐一取出每張圖檢測結果的每個檢測資訊(每組骨架17組座標及bbox左上、右下角點座標-xyxy)
    for item in result:
        image_data.append({
            "bbox": item.bboxes.xyxy[0].tolist(),
            "keypoints": item.keypoints.xy[0].tolist()
        })
    results_data[image_name] = image_data
with open("results.json", "w") as f:
    json.dump(results_data, f, indent=4)
```



# Hint: How to load the result

- Read the result file

```
with open('results.json', 'r') as file:  
    data = json.load(file)
```

- Show the first detection bbox of image1.jpg

```
print(data['image1.jpg'][0]['bbox'])
```

- Iterate through and display all bbox

```
for img_name, detections in data.items():  
    for detection in detections:  
        print(img_name, ": ", detection["bbox"])
```

# Application: Forehand and backhand determination

- **Using the predicted skeleton** to classify the player's stroke as forehand or backhand(正手或反手擊球).



There are typically clear indicators, such as **the position of the hands or legs on the opposite side of the body**, that distinguish between forehand and backhand strokes. (Since we will focus on data that is easily classifiable, rule-based methods like **if-else** can be utilized for classification)

# Application: The output format

- Output the forehand/backhand results to the terminal. Basic format:

```
01.jpg : Bottom Half player - Forehand  
01.jpg : Top Half player - Forehand  
02.jpg : Bottom Half player - Forehand
```

- Due to insufficient information to determine who is hitting the shuttle, please evaluate the action of **all players** and classify it as either a forehand or backhand stroke.
- The program `player_half_court_classifier.py` helps you determine the players' positions on the court. Please download and refer to the following method for use.

```
from player_half_court_classifier import classify_player  
player_bbox_xyxy = [389.0, 384.0, 476.0, 654.0]  
  
# The bbox should be list and the coordinates of the upper-  
# left and lower-right corners of the bounding box  
print(classify_player(player_bbox_xyxy)) # Bottom Half
```

# Demo

- We will prepare a folder with 10 photos. Use your model to predict and save the results in results.json (format as defined on page 19).
- Then, read the results.json and use your classification program to determine whether each player's action is a forehand or backhand stroke, displaying the results on the screen.
- We will check:
  - Whether the model is trained **based on yolov8n-pose** (other pre-trained models cannot be used).
  - The similarity between the output and ground truth (evaluated using OKS).
  - Whether the classification of the player's forehand/backhand stroke is accurate.





# Report

- Please hand in a report explaining the methods you used to improve the pose estimation(if you did so) and classify the strokes as forehand or backhand.



# Grading

- Data Label (20%)
- Successfully executable (30%)
- Report(20%)
- Performance(30%)



# Deadline

- Data Label
  - On 12/9(一) 23:59
- Report
  - On 12/16(一) 23:59
- Demo
  - On 12/17(二)
- Each group needs at least 1 member to demonstrate
- Late Submission
  - 10% less per day (e.g. 90->81->73)
- If you have any problems about submission & demo please contact TAs before deadline



# TAs

- [yqliiii.cs12@nycu.edu.tw](mailto:yqliiii.cs12@nycu.edu.tw)
- [ytdeng.cs13@nycu.edu.tw](mailto:ytdeng.cs13@nycu.edu.tw)