1. Swimmer를 뺀 나머지 category를 삭제하는 방향으로 실습

1. Coco.json에서 swimmer 카테고리, 이미지만 남겨주기

   -카테고리는 수동으로 삭제

```
import json

# 원본 JSON 파일 경로
input_file_path = "./data/train/_annotations.coco.json"

# 수정된 JSON 파일 경로
output_file_path = "./data/train/_annotations.coco.json"

# 수정할 카테고리 이름
target_category_name = "swimmer"

# JSON 파일 불러오기
with open(input_file_path, "r") as json_file:
    data = json.load(json_file)

# "swimmer" 카테고리의 ID 찾기
swimmer_category_id = next(category["id"] for category in data["categories"] if
category["name"] == target_category_name)

# "images" 항목에서 "swimmer" 카테고리에 해당하는 이미지만 남기기
swimmer_image_ids = set()
for annotation in data["annotations"]:
    if annotation["category_id"] == swimmer_category_id:
        swimmer_image_ids.add(annotation["image_id"])

# "images" 정보와 "annotations" 정보 수정
filtered_images = [image for image in data["images"] if image["id"] in swimmer_image_ids]
filtered_annotations = [annotation for annotation in data["annotations"] if
annotation["category_id"] == swimmer_category_id]

data["images"] = filtered_images
data["annotations"] = filtered_annotations

# 수정된 데이터 저장
with open(output_file_path, "w") as json_file:
    json.dump(data, json_file)
```

2. 주어진 coco 데이터 yolo 포맷으로 변경하기

- Yolov8 git clone 하고 그안에 데이터셋과 cocotoyolo.py 파일 만들기
- train/ valid 각각 돌려주기

```
- import json
```

```
import os
import shutil

"""
"annotations":
[{"id":0,"image_id":0,"category_id":1,"bbox":[890,979,21.5,62],"area":1333,"segmentation":[
],"iscrowd":0},
{"id":1,"image_id":0,"category_id":1,"bbox":[626,886,59.5,43.5],"area":2588.25,"segmentati
on":[],"iscrowd":0},
{"id":2,"image_id":0,"category_id":1,"bbox":[842,602,45.5,30.5],"area":1387.75,"segmentati
on":[],"iscrowd":0},
{"id":3,"image_id":0,"category_id":1,"bbox":[690,398,25.5,39.5],"area":1007.25,"segmentati
on":[],"iscrowd":0},
{"id":4,"image_id":0,"category_id":1,"bbox":[346,292,27.5,23.5],"area":646.25,"segmentatio
n":[],"iscrowd":0},
"id":5,"image_id":0,"category_id":1,"bbox":[178,216,31.5,36],"area":1134,"segmentation":[],
"iscrowd":0},
{"id":6,"image_id":0,"category_id":1,"bbox":[31,672,54,30],"area":1620,"segmentation":[],"is
crowd":0},
{"id":7,"image_id":0,"category_id":1,"bbox":[171,1079,54,32],"area":1728,"segmentation":[],
"iscrowd":0},


"""

# yolo dataset create folder
os.makedirs("ultralytics/cfg/swim_yolo_data/train/images/", exist_ok=True)
os.makedirs("ultralytics/cfg/swim_yolo_data/train/labels/", exist_ok=True)
os.makedirs("ultralytics/cfg/swim_yolo_data/val/images/", exist_ok=True)
os.makedirs("ultralytics/cfg/swim_yolo_data/val/labels/", exist_ok=True)

# image path
image_path = "./data/valid/"

# COCO annotation file path
coco_annotation_path = './data/valid/_annotations.coco.json'

# YOLO format annotation save folder
yolo_annotation_folder = './ultralytics/cfg/swim_yolo_data/val/labels/'

# YOLO format image save folder
yolo_image_folder = './ultralytics/cfg/swim_yolo_data/val/images/'

# COCO class names (modify according to your dataset)
coco_classes = ['swimmer']
yolo_Classes = {'swimmer':0}

# Load COCO annotations
with open(coco_annotation_path, 'r') as f:
    coco_annotations = json.load(f)

image_infos = coco_annotations['images']
ann_infos = coco_annotations['annotations']

for image_info in image_infos:
    image_file_name = image_info['file_name']
    file_name = image_file_name.replace(".jpg", "")
    id = image_info['id']
    image_width = image_info['width']
```

```python
        image_height = image_info['height']

        for ann_info in ann_infos:
            if ann_info['image_id'] == id:
                # coco -> two stage 1, yolo -> one stage 0
                category_id = ann_info['category_id'] - 1
                x, y, w, h = ann_info['bbox']

                # xywh -> center x center y w h
                x_center = (x + w / 2) / image_width
                y_center = (y + h / 2) / image_height
                w /= image_width
                h /= image_height

                # Copy image to YOLO image folder
                source_image_path = os.path.join(image_path, image_file_name)
                destination_image_path = os.path.join(yolo_image_folder, image_file_name)
                shutil.copy(source_image_path, destination_image_path)

                # Write annotation to text file
                yolo_line = f"{category_id} {x_center:.6f} {y_center:.6f} {w:.6f} {h:.6f}\n"
                text_path = os.path.join(yolo_annotation_folder, f"{file_name}.txt")
                with open(text_path, 'a') as f:
                    f.write(yolo_line)
-
```

3. Swimmer.yaml .yaml 파일 생성

```yaml
train: ../swim_yolo_data/train/images   # train images (relative to 'path') 128 images
val: ../swim_yolo_data/val/images   # val images (relative to 'path') 128 images


# Classes
names:
  0: swimmer
```

4. Train.py 파일 만들고 훈련시키기

```python
from ultralytics import YOLO

# train model info
model = YOLO('yolov8s.pt')
if __name__ =='__main__':
    model.train(data='sm.yaml', epochs=100, batch=34, degrees=5, lrf=0.025)
```

데이터 양이 너무 적어짐

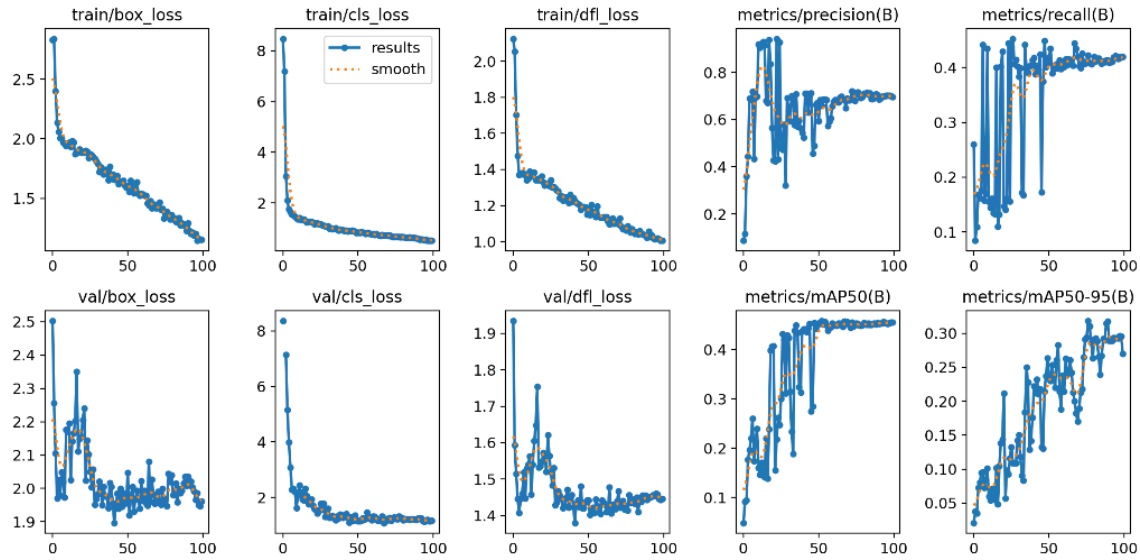
```
    len_cls, len_boxes, len_segments = (sum(x) for x in zip(*lengths))
    ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
ValueError: not enough values to unpack (expected 3, got 0)
```

2. 카테고리를 최대한 통일시키는 방식으로 수정

annotation에서 삭제한 항목 : aerial-pool, umpire

```
import json

data_path = "./data/train/_annotations.coco.json"

# 원본 JSON 데이터 로드
with open(data_path, 'r') as json_file:
    data = json.load(json_file)

# 삭제할 카테고리 이름
categories_to_remove = ["aerial-pool", "umpire"]

# 삭제할 카테고리의 id 확인
categories_to_remove_ids = []
for category in data["categories"]:
    if category["name"] in categories_to_remove:
        categories_to_remove_ids.append(category["id"])

# 카테고리 삭제
data["categories"] = [category for category in data["categories"] if category["id"] not in
categories_to_remove_ids]

# 해당 카테고리가 속한 이미지의 id 확인
images_to_remove_ids = set()
for annotation in data["annotations"]:
    if annotation["category_id"] in categories_to_remove_ids:
        images_to_remove_ids.add(annotation["image_id"])

# 이미지 및 어노테이션 삭제
data["images"] = [image for image in data["images"] if image["id"] not in images_to_remove_ids]
data["annotations"] = [annotation for annotation in data["annotations"] if annotation["image_id"] not
in images_to_remove_ids]

# 카테고리 id 순차적으로 재조정
id_mapping = {}
new_id = 0
for category in data["categories"]:
    if category["id"] not in categories_to_remove_ids:
        id_mapping[category["id"]] = new_id
        category["id"] = new_id
        new_id += 1

# 어노테이션의 카테고리 id 수정
for annotation in data["annotations"]:
    annotation["category_id"] = id_mapping[annotation["category_id"]]

# 수정된 데이터를 새로운 파일에 저장
updated_data_path = "./data/train/updated_annotations.coco.json"
with open(updated_data_path, 'w') as json_file:
    json.dump(data, json_file, indent=4)
```

Coco to yolo, yaml 파일도 수정하여 train 진행



Tracking 코드에서 tracking_id에 대한 부분(수업 코드) 삭제하여 모든 사람 트래킹하도록 수정

```
import cv2
import numpy as np
from ultralytics import YOLO
from collections import defaultdict

# Load the yolov8 model
model = YOLO('./runs/detect/train/weights/best.pt')

# open the video file
video_path = "./swimming_pool.mp4"
cap = cv2.VideoCapture(video_path)

# Store the track history for each person
track_histories = defaultdict(lambda: [])

while cap.isOpened():
    # Read a frame from video
    success, frame = cap.read()

    if success:
        # Run yolov8 tracking on the frame, persisting tracks between frames
        results = model.track(frame, persist=True)

        for result in results:
            boxes = result.boxes.xyxy.cpu().tolist()
            track_ids = result.boxes.id.int().cpu().tolist()

            for box, track_id in zip(boxes, track_ids):
                x1, y1, x2, y2 = box
                track = track_histories[track_id]

                # Append the box coordinates to the track history
                track.append((float(x1), float(y1), float(x2), float(y2)))
                if len(track) > 30:
                    track.pop(0)    # retain 30 tracks for 30 frames
```

```
                    # Draw the tracking bounding box
                    img = cv2.rectangle(frame, (int(x1), int(y1)), (int(x2), int(y2)), (0, 255, 0), 2)

            # Display the annotated frame
            cv2.imshow("Tracking..", img)

            if cv2.waitKey(1) & 0xFF == ord('q'):
                break
        else:
            break

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