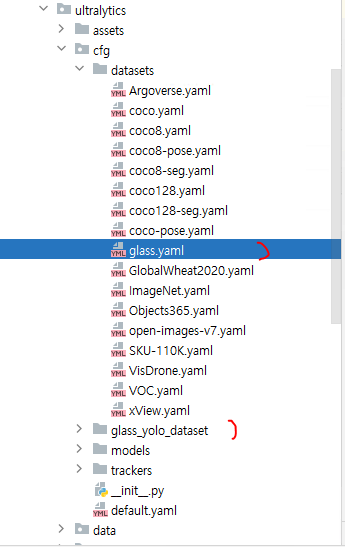
1. 주어진 coco 데이터 yolo 포맷으로 변경하기

Yolov8 git clone 하고 그안에 데이터셋과 cocotoyolo.py 파일 만들기 // train/ valid 각각 해줘야하니 이부분만 코드 수정해서 돌리기

|  |
| --- |
| 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,"segmentation":[],"iscrowd":0}, {"id":2,"image\_id":0,"category\_id":1,"bbox":[842,602,45.5,30.5],"area":1387.75,"segmentation":[],"iscrowd":0}, {"id":3,"image\_id":0,"category\_id":1,"bbox":[690,398,25.5,39.5],"area":1007.25,"segmentation":[],"iscrowd":0}, {"id":4,"image\_id":0,"category\_id":1,"bbox":[346,292,27.5,23.5],"area":646.25,"segmentation":[],"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":[],"iscrowd":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("./glass\_yolo\_data/train/images/", exist\_ok=True) os.makedirs("./glass\_yolo\_data/train/labels/", exist\_ok=True) os.makedirs("./glass\_yolo\_data/val/images/", exist\_ok=True) os.makedirs("./glass\_yolo\_data/val/labels/", exist\_ok=True)  *# image path* image\_path = "./glass\_dataset/valid/"  *# COCO annotation file path* coco\_annotation\_path = './glass\_dataset/valid/\_annotations.coco.json'  *# YOLO format annotation save folder* yolo\_annotation\_folder = './glass\_yolo\_data/val/labels/'  *# YOLO format image save folder* yolo\_image\_folder = './glass\_yolo\_data/val/images/'  *# COCO class names (modify according to your dataset)* coco\_classes = ['glass'] yolo\_Classes = {'glass': 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']  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  *# image copy to dst folder  # 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 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) |

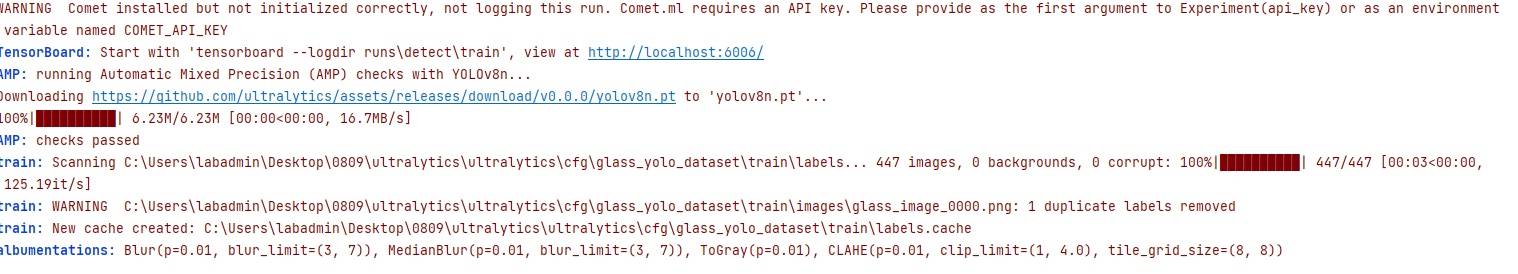
1. 만들어진 glass\_yolo\_dataset 를 cfg 폴더안에 넣고 cfg > datasets에서 yaml 파일 하나 복사해서 glass.yaml 파일 생성



|  |
| --- |
| train: ../glass\_yolo\_data/train/images *# train images (relative to 'path') 128 images* val: ../glass\_yolo\_data/val/images *# val images (relative to 'path') 128 images   # Classes* names:  0: glass |

1. Ultralytics에 train.py 만들기

|  |
| --- |
| from ultralytics import YOLO  *# train model info* model = YOLO('yolov8s.pt') if \_\_name\_\_ =='\_\_main\_\_':  model.train(data='glass.yaml', epochs=100, batch=34, degrees=5, lrf=0.025) |



훈련 시작이 됩니다. 훈련 끝나면 run가서 결과 확인

**Xml 문서 저장**

|  |
| --- |
| *#### cvat* import os import glob import cv2 import xml.etree.ElementTree as ET from ultralytics import YOLO """ <annotations>  <image id="0" name="IMG\_4913\_JPG\_jpg.rf.4f67c223e9cbf0ed07236bfe142aaaee.jpg" width="1920" height="1080">  <box label="bond\_1" source="manual" occluded="0" xtl="1026.81" ytl="324.65" xbr="1309.74" ybr="479.46" z\_order="0"> </box>  </image>  <image id="1" name="IMG\_4913\_JPG\_jpg.rf.4f67c223e9cbf0ed07236bfe142aaaee.jpg" width="1920" height="1080">  <box label="bond\_1" source="manual" occluded="0" xtl="1026.81" ytl="324.65" xbr="1309.74" ybr="479.46" z\_order="0"> </box>  </image> </annotations>  """ *# model load* model = YOLO("./runs/detect/train4/weights/best.pt") data\_path = "./glass\_dataset/test/" data\_path\_list = glob.glob(os.path.join(data\_path, "\*.jpg"))  tree = ET.ElementTree() root = ET.Element("annotations") """ 1.  <annotations> </annotations> """ id\_number = 0 xml\_path = "./test.xml"  for path in data\_path\_list :  names = model.names  results = model.predict(path, save=False, imgsz=640, conf=0.5)  boxes = results[0].boxes  box\_info = boxes  box\_xyxy = box\_info.xyxy  cls = box\_info.cls  image = cv2.imread(path)  img\_height, img\_width, \_ = image.shape  file\_name = os.path.basename(path)  print(file\_name, img\_height, img\_width)   xml\_frame = ET.SubElement(root, "image", id="%d" % id\_number, name=file\_name, width="%d" % img\_width, height="%d" % img\_height)  """  2.   <annotations>  <image id="0" name="IMG\_4913\_JPG\_jpg.rf.4f67c223e9cbf0ed07236bfe142aaaee.jpg" width="1920" height="1080">  </annotations>  """  for bbox, class\_number in zip(box\_xyxy, cls) :  class\_number = int(class\_number.item())  class\_name\_temp = names[class\_number]  print(class\_name\_temp)  """  3.   <annotations>  <image id="0" name="IMG\_4913\_JPG\_jpg.rf.4f67c223e9cbf0ed07236bfe142aaaee.jpg" width="1920" height="1080">  <box label="bond\_1" source="manual" occluded="0" xtl="1026.81" ytl="324.65" xbr="1309.74" ybr="479.46" z\_order="0"> </box>  </annotations>  """  x1 = int(bbox[0].item())  y1 = int(bbox[1].item())  x2 = int(bbox[2].item())  y2 = int(bbox[3].item())  ET.SubElement(xml\_frame, "box", label=str(class\_name\_temp), source="manual",occluded="0",  xtl=str(x1), ytl=str(y1), xbr=str(x2), ybr=str(y2), z\_order="0")   id\_number +=1  tree.\_setroot(root)  tree.write(xml\_path, encoding="utf-8") |

이제 CVAT가서 오토 라벨링 해보자

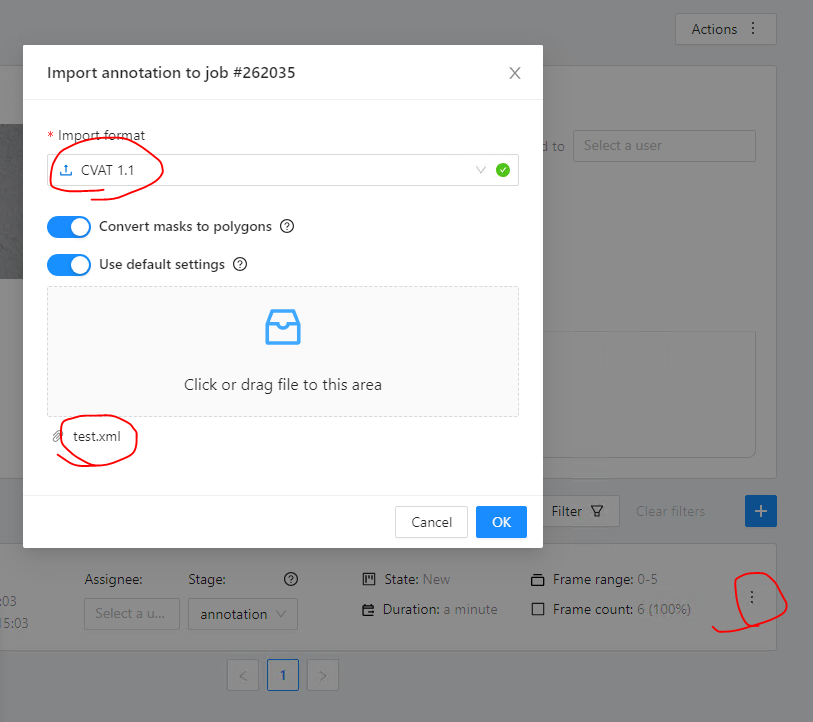
<https://www.cvat.ai/> -> TRY FOR FREE 한다음에 프로젝트 생성하고

name 지정

add label 해주고 콘티뉴-raw 보면 생성된거 볼 수 있다.

이미지 다 드래그 드롭해서 넣어준 후 서브밋 앤 오픈 해준다

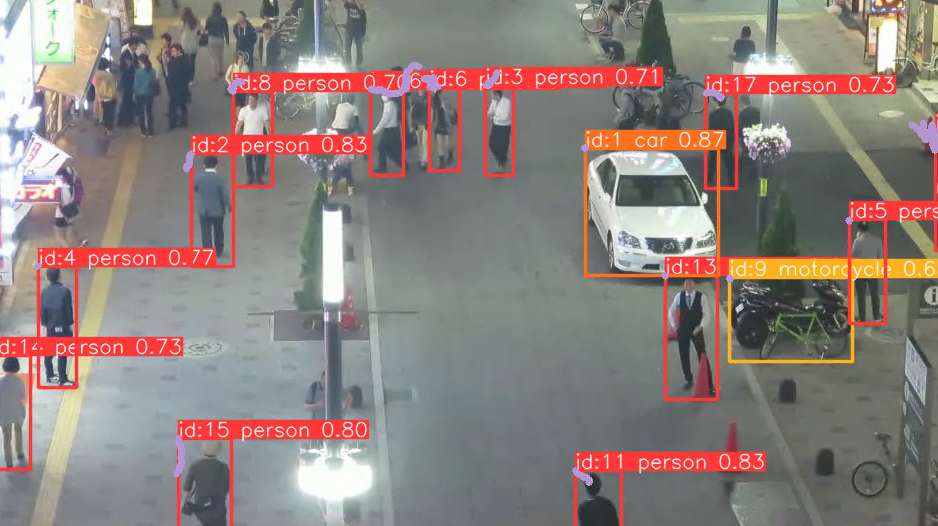
그후 import annotation 해주고 cvat 선택후 xml넣어준다



Yolo tracking

Tracking할 비디오 파일 다운로드

|  |
| --- |
| import cv2 import numpy as np from ultralytics import YOLO from collections import defaultdict  *# Load the yolov8 model* model = YOLO('yolov8n.pt')  *# open the video file* video\_path = "./MOT17-04-SDP-raw.webm" cap = cv2.VideoCapture(video\_path) *#0으로 하면 웹캠  # Store the track history* track\_history = defaultdict(lambda : [])  while cap.isOpened():  *# Read a frame from video* success, frame = cap.read()  *#print(frame) # 영상 잘 나오는지 확인* if success :   *# run yolov8 tracking on the frame, persisting tracks between frames* results = model.track(frame, persist=True)   *#print(results) # 박스정보 등 잡기* boxes = results[0].boxes.xyxy.cpu().tolist(   )  track\_ids = results[0].boxes.id.int().cpu().tolist()   annotated\_frame = results[0].plot()   print(track\_ids) *# 박스정보* for box, track\_id in zip(boxes, track\_ids):  x1, y1, x2, y2 = box  track = track\_history[track\_id]   track.append((float(x1), float(y1)))  if len(track) > 30 :  track.pop(0) *# retain 90 tracks for 90 frames   # Drew the tracking lines* points = np.hstack(track).astype(np.int32).reshape((-1,1,2))  cv2.polylines(annotated\_frame, [points], isClosed=False, color=(210,150,180), thickness=5)    *# Display the annotated frame* cv2.imshow('Tracking...', annotated\_frame)   if cv2.waitKey(30) & 0xFF == ord('q') :  break  else :  break |



1. 데이터 제공(coco 양식 데이터)

-해당 라벨이 어떤씩에 라벨이 되어있는지 체크 필요합니다

시각화

1. 학습 모델 mmdetection, yolov8
2. 학습 결과/tracking 해보기

* 테스트 할 수 있는 비디오 영상 제공