메인보드 부품 가이드 라인

Deep Learning - Object Detection

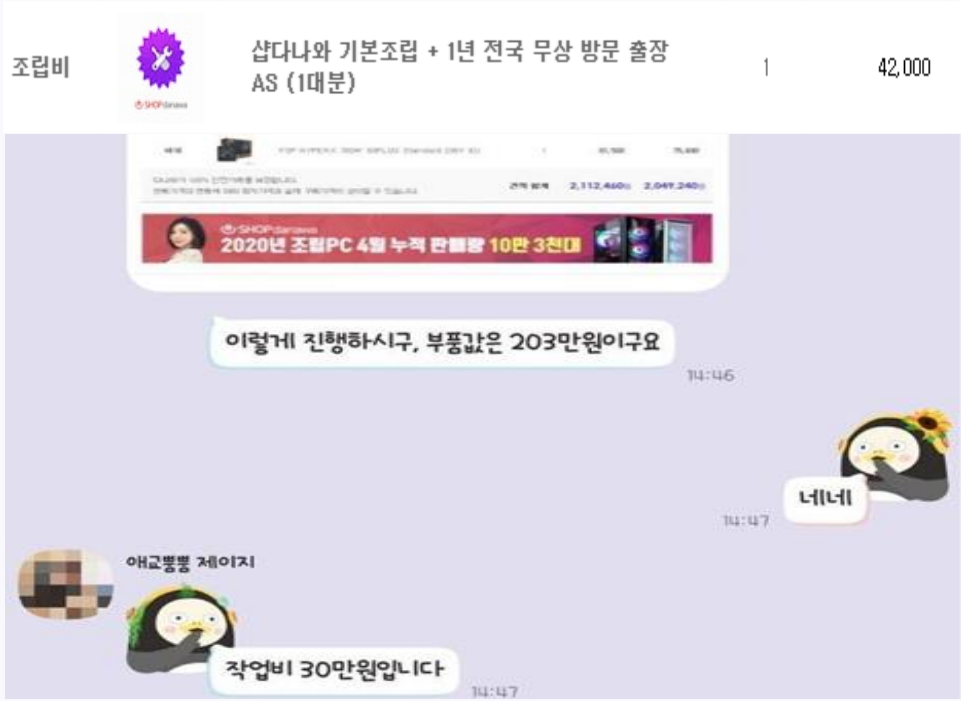
Project subject

다나와 조립비가 6만원이나 되었네요... > 자유게시판

2023. 3. 5. — 조립하는사람 1~2시간 쓰는임금이 6만원쯤이면 싼거 아닌가요? 선정리 다해주고 테스트 다해주고 초기불량 대응 빨리하고.

김나성 - 다나와 조립비 진짜 고민되네요

2022. 11. 24. — 20년도에 살때는 2.5만원인가했던거같은데. 왜 지금보니 5만원인지.. 조립대행 맡기는게 편하긴한데.. 5만원은좀 많이 고민하게되는액수군요.



Project Procedure

Crawling
- 다나와 사이트 크롤링
- 이미지 수집

CVAT(Label)
- 크롤링 이미지 데이터 기반으로 라벨링 진행

Object Detection
- 라벨링 데이터 기반 이미지, 영상 Object Detection 진행

Streamlit
- Streamlit을 이용, 웹페이지에 Object Detection 구현

Team Member

김 기 준

- 전체 일정 조율 및 역할 분배
- 다나와 사이트 크롤링 협업
- 이미지 데이터 CVAT을 통한 Label작업 협업
- Label 작업 후 이미지, 동영상 딥러닝 실행 및 결과 도출
- PPT 제작

김 태 환

- 다나와 사이트 크롤링
- 이미지 데이터 수집
- 이미지 데이터 CVAT을 통한 Label작업
- Streamlit을 이용한 웹 페이지 구현 협업

문 창 주

- 다나와 사이트 크롤링 협업
- 이미지 데이터 CVAT을 통한 Label작업 협업
- Streamlit을 이용한 웹 페이지 구현
- PPT 제작 협엄

Project Schedule

SUN	MON	TUE	WED	THU	FRI	SAT
	26	27	28	29	1	2
	- 주제 선정 및 역할 분담 - 크롤링 시작		- 크롤링 마무리 - 테스트 이미지 딥러닝 실행 및 결과 확인	- CVAT을 통해 이미지 라벨링 작업 진행 - 라벨링 데이터 딥러닝 실행 및 결과 확인		- 동영상 데이터 딥러닝 실행 및 결과 확인
3	4	5				
- Streamlit 을 통해 웹페이지 구축 - PPT 제작		발표				

데이터 수집(Crawling)

```
# 크롤링 코드
import requests
from bs4 import BeautifulSoup
from selenium import webdriver
from selenium.webdriver.common.by import By
from selenium.webdriver.common.keys import Keys
import time
from urllib.request import urlretrieve
from PIL import Image
from selenium.webdriver.common.action_chains import ActionChains


list_num = 0

# 다나와 접속
url = 'https://www.danawa.com/'

driver = webdriver.Chrome()
print("\r다나와 접속 시작.", end="")
driver.get(url)
time.sleep(1)

key = '메인보드'

elem = driver.find_element(By.CSS_SELECTOR, "#AKCSearch")
print("\r메인보드로 검색 시작.")
elem.send_keys(key)
elem.send_keys(Keys.RETURN)
print("\r검색 완료.")
time.sleep(2)
```

데이터 수집(Crawling)

```
# 끝페이지 탐색

find_end = True
comeback = True
end = 0

print('\r끝 페이지 탐색 시작.')

while find_end:
    try:
        flip_right = driver.find_element(By.CSS_SELECTOR, f"div.prod_num_nav > div.num_nav_wrap > a.nav_next")
        flip_right.click()
        time.sleep(2)
    except:
        total_end = driver.find_element(By.CSS_SELECTOR,'.number_wrap > a:last-child').text
        end = int(total_end)
        find_end = False
        break

  while comeback:
      try:
          flip_left = driver.find_element(By.CSS_SELECTOR, f"div.prod_num_nav > div.num_nav_wrap > a.nav_prev")
          flip_left.click()
          time.sleep(1)
      except:
          start_number = driver.find_element(By.CSS_SELECTOR,'.number_wrap > a:first-child')
          start_number.click()
          comeback = False
          print('\r처음으로 복귀 완료.')
          break

  print('='*20)

  time.sleep(2)
```

데이터 수집(Crawling)

```
# 최하단 페이지 이동 컨트롤 시 필요한 변수
page_ctrl = 1

# 크롤링 시작
for idx in range(1,end + 1):
    print(f'{idx} 페이지 진행중...')

    for num in range(1,40):
        try:
            # 주소
            thumbnail = driver.find_element(By.CSS_SELECTOR, f"#productListArea > div.main_prodlist > ul.product_list > li.width_change:nth-child({num}) > div > div.thu
            thumb_btn = driver.find_element(By.CSS_SELECTOR, f"#productListArea > div.main_prodlist > ul.product_list > li.width_change:nth-child({num}) > div > div.thu

            # 호버한 뒤 클릭
            ActionChains(driver).move_to_element(thumbnail).perform()
            ActionChains(driver).click(thumb_btn).perform()

            time.sleep(1)

            for i in range(1,5):
                print(f'\r{num}번째 품목 이미지 다운로드 시작', end = "")
                try:
                    # 이동
                    if i == 4:
                        print(f'\r{num}번째 품목 이미지 다운로드 완료', end = "")
                        elem = driver.find_element(By.CSS_SELECTOR, "#closeImgExpandLayer")
                        elem.click()
                        time.sleep(0.5)
```

데이터 수집(Crawling)

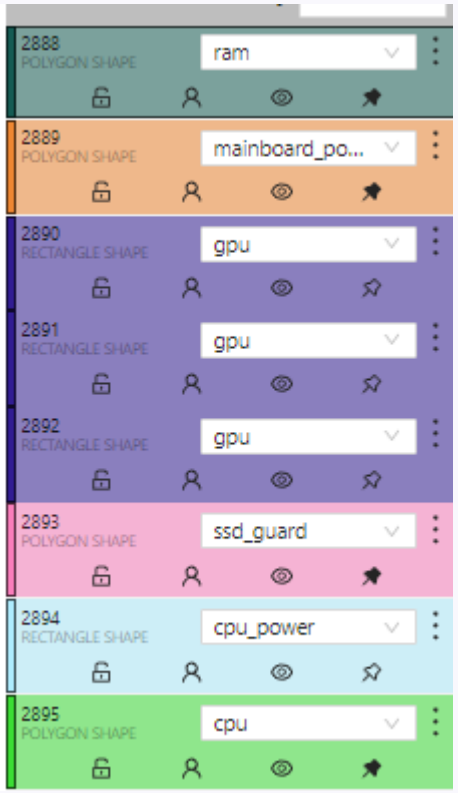
```
                else:
                    try:
                        elem = driver.find_element(By.CSS_SELECTOR, f"#imageThumbNail_{i}")
                        elem.click()
                        time.sleep(0.5)

                        # 선택
                        elem = driver.find_element(By.CSS_SELECTOR, "#expandViewLayer > img")
                        img_url = elem.get_attribute('src')
                        time.sleep(0.5)

                        # 저장
                        file_name = f"origin/mainboard_{idx}_{num}_{i}.jpg"
                        urlretrieve(img_url, file_name)
                        time.sleep(0.5)
                    except:
                        pass
            except:
                print(f'\r{num}번째 품목 이미지 다운로드 완료', end = "")
                elem = driver.find_element(By.CSS_SELECTOR, "#closeImgExpandLayer")
                elem.click()

    except:
        pass

print()
```

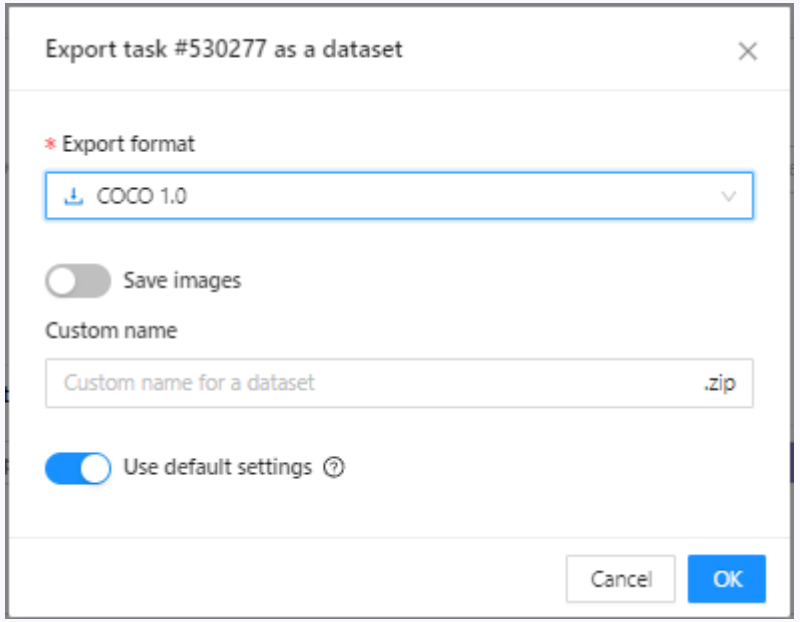
데이터 수집(Crawling)

```
    # 마지막 페이지 확인
    is_page_last = driver.find_element(By.CSS_SELECTOR, f".prod_num_nav > div > div > a:last-child").text

    if int(idx) % 10 == 0:
        print(f'{idx} 페이지 마지막에 도달', end="")

        flip_right = driver.find_element(By.CSS_SELECTOR, f"div.prod_num_nav > div.num_nav_wrap > a.nav_next")
        print(f'\r{idx + 1} 페이지로 넘어갑니다.')
        print("="*20)
        page_ctrl = 1
        flip_right.click()

        time.sleep(1)

    else:
        print(f'{idx} 페이지 마지막.', end="")
        page_nav = driver.find_element(By.CSS_SELECTOR, f".prod_num_nav > div > div > a:nth-child({page_ctrl + 1})")

        print(f'\r{idx + 1}페이지로 넘어갑니다.')
        print("="*20)
        page_ctrl += 1
        page_nav.click()
        time.sleep(1.5)

print(f'총 {idx} 페이지 스크랩핑 마무리')
print("전 품목 이미지 다운로드 완료")
```

CVAT


CPU 2895 (MANUAL)


2888
POLYGON SHAPE
ram
2889
POLYGON SHAPE
mainboard_po...
2890
RECTANGLE SHAPE
gpu
2891
RECTANGLE SHAPE
gpu
2892
RECTANGLE SHAPE
gpu
2893
POLYGON SHAPE
ssd_guard
2894
RECTANGLE SHAPE
cpu_power
2895
POLYGON SHAPE
cpu

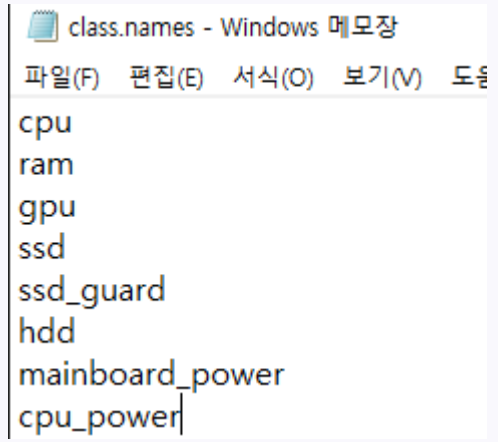

Export task #530277 as a dataset
* Export format
COCO 1.0
Save images
Custom name
Custom name for a dataset
.zip
Use default settings
Cancel
OK

Object Detection

Label(COCO data set)


```
{"licenses":[{"name":"","id":0,"url":""}],"info":{"contributor":"","date_created":"","description":"","url":"",
"version":"","year":""},"categories":[{"id":1,"name":"cpu","supercategory":""},{"id":2,"name":"ram",
"supercategory":""},{"id":3,"name":"gpu","supercategory":""},{"id":4,"name":"ssd","supercategory":""},{"id":5,
"name":"ssd_guard","supercategory":""},{"id":6,"name":"hdd","supercategory":""},{"id":7,
"name":"mainboard_power","supercategory":""},{"id":8,"name":"cpu_power","supercategory":""}],"images":[{"id":1,
"width":500,"height":500,"file_name":"mainboard (1).jpg","license":0,"flickr_url":"","coco_url":"",
"date_captured":0},{"id":2,"width":480,"height":480,"file_name":"mainboard (10).jpg","license":0,
"flickr_url":"","coco_url":"","date_captured":0},{"id":3,"width":500,"height":500,"file_name":"mainboard (100).
jpg","license":0,"flickr_url":"","coco_url":"","date_captured":0},{"id":4,"width":500,"height":500,
"file_name":"mainboard (1000).jpg","license":0,"flickr_url":"","coco_url":"","date_captured":0},{"id":5,
"width":500,"height":500,"file_name":"mainboard (1002).jpg","license":0,"flickr_url":"","coco_url":"",
```


Class



Yaml

```yaml
path: /content/drive/MyDrive/dl_pj2/mainboard/data
train: ./images/train
val: ./images/val
test: ./images/test

nc: 8
names: ['cpu','ram','gpu','ssd','ssd_guard','hdd','mainboard_power','cpu_power']
```

Object Detection

COCO data set

```
!python example.py --datasets COCO --img_path ../mainboard/data/images --label ../mainboard/labels.json ₩
--convert_output_path ../mainboard/data/labels --img_type ".jpg" --manifest_path ./ --cls_list_file ../mainboard/class.names
```

```
COCO Parsing:   |████████████████████████████████████████| 100.0% (8845/8845)  Complete

YOLO Generating:|████████████████████████████████████████| 100.0% (1133/1133)  Complete

YOLO Saving:    |████████████████████████████████████████| 100.0% (1133/1133)  Complete
```

Object Detection

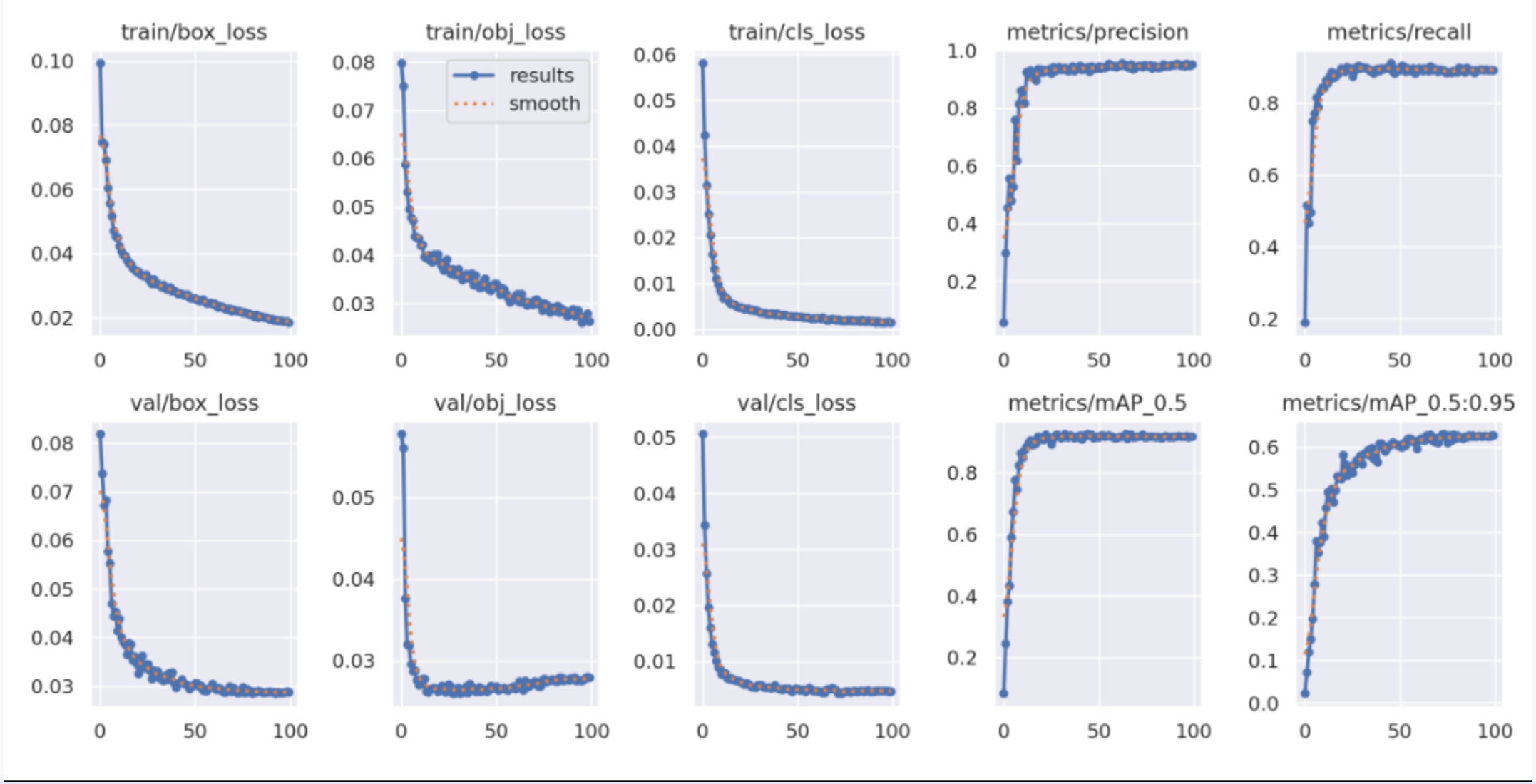
간단한 모델 성능 평가

```
# 모델의 성능 평가 - object detection 성능 평가

!python val.py --data ../mainboard/custom.yaml --weights runs/train/exp5/weights/best.pt
```

```
val: data=../mainboard/custom.yaml, weights=['runs/train/exp5/weights/best.pt'], batch_size=32, imgsz=640, conf_t
YOLOv5 🚀 v7.0-287-g574331f9 Python-3.10.12 torch-2.1.0+cu121 CUDA:0 (Tesla T4, 15102MiB)

Fusing layers...
Model summary: 212 layers, 20881221 parameters, 0 gradients, 47.9 GFLOPs
val: Scanning /content/drive/MyDrive/dl_pj2/mainboard/data/labels/val... 339 images, 0 backgrounds, 0 corrupt: 10
val: New cache created: /content/drive/MyDrive/dl_pj2/mainboard/data/labels/val.cache
                 Class     Images  Instances          P          R      mAP50   mAP50-95: 100% 11/11 [00:18<00:00,
                   all        339       2790      0.944      0.893      0.923      0.631
                   cpu        339        336      0.998          1      0.995      0.878
                   ram        339        376      0.981      0.967      0.982      0.778
                   gpu        339        788      0.866      0.834      0.805      0.515
                   ssd        339        274      0.933      0.796      0.879      0.476
             ssd_guard        339        183      0.945      0.869      0.905      0.663
                   hdd        339        204      0.897      0.852       0.93      0.633
       mainboard_power        339        322       0.99       0.95      0.971      0.538
             cpu_power        339        307      0.944      0.875      0.919      0.567
Speed: 0.4ms pre-process, 16.9ms inference, 11.2ms NMS per image at shape (32, 3, 640, 640)
Results saved to runs/val/exp4
```

Object Detection

실제 이미지 평가



Object Detection

실제 이미지 평가

Object Detection

실제 영상 평가

Streamlit

세팅

```
import numpy as np

import streamlit as st
from PIL import Image

import time
import os
import torch
import sys
import pathlib
temp = pathlib.PosixPath
pathlib.PosixPath = pathlib.WindowsPath

from moviepy.editor import VideoFileClip
import torchvision.transforms as transforms


from moviepy.video.io.VideoFileClip import VideoFileClip
from moviepy.video.io.ImageSequenceClip import ImageSequenceClip

import tempfile
import cv2


from moviepy.editor import concatenate_videoclips, VideoClip
from moviepy.video.io.ffmpeg_writer import FFMPEG_VideoWriter
```

Streamlit

Streamlit

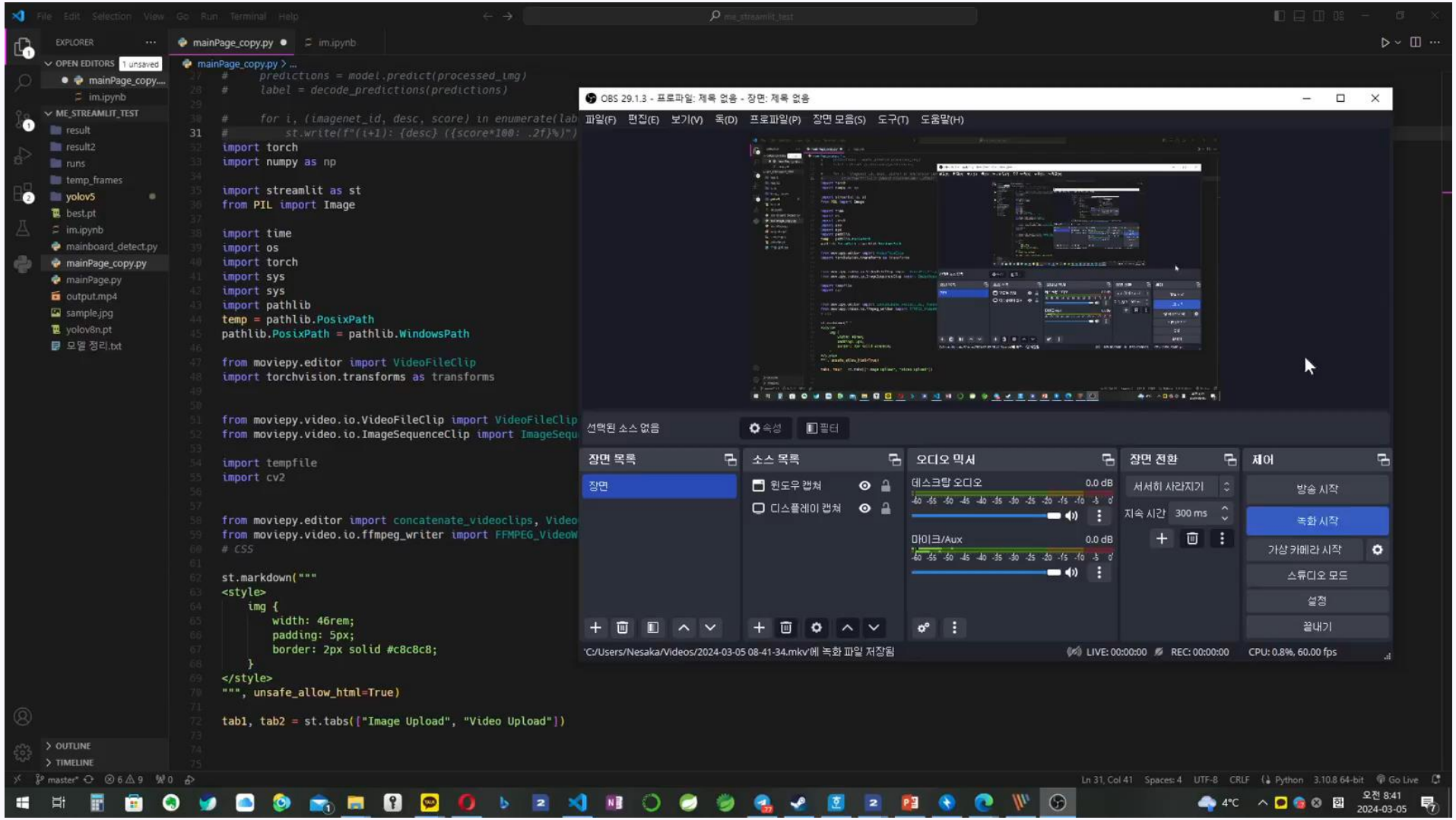
```
st.markdown("""
<style>
    img {
        width: 46rem;
        padding: 5px;
        border: 2px solid #c8c8c8;
    }
</style>
""", unsafe_allow_html=True)

tab1, tab2 = st.tabs(["Image Upload", "Video Upload"])



# 첫 번째 탭의 내용
with tab1:
    st.title('Mainboard parts Detect Tool',)

    uploaded_file = st.file_uploader('Please Upload files in this area', type=['jpg', 'jpeg', 'png'])
                                                                          #'jpg', 'jpeg', 'png','avi','mp4', 'mkv', 'wav'
```

Streamlit

```python
if uploaded_file is not None:
    image = Image.open(uploaded_file)
    st.image(image, use_column_width=True) #caption='Uploaded Image'

    #####
    sys.path.insert(0, 'yolov5')

    sys.path.append('yolov5')  # YOLOv5 폴더 경로로 변경

    # 'custom' 모델 로드를 위한 hubconf.py 내의 함수 사용
    # 'path/to/best.pt'를 모델 파일의 실제 경로로 변경
    model = torch.hub.load('yolov5', 'custom', path='best.pt', source='local')

    # 모델을 평가 모드로 설정
    model.eval()

    temp = model(image)

    result = temp.save()

    re_img = Image.open('runs/detect/exp/image0.jpg')

    st.image(re_img)

    time.sleep(0.1)
    os.remove('runs/detect/exp/image0.jpg')
    time.sleep(0.1)
    os.rmdir('runs/detect/exp/')
```

Streamlit

실제 시연 영상 (이미지)



소감

“이미지 라벨링 작업 시 데이터 수, 라벨 종류가 많았기에 생각보다 많은 시간을 할애했다. 그로 인해 전체적으로 다른 작업에 시간 할당을 충분히 하지 못하여 아쉬웠다. 이번 경험 덕분에 프로젝트 시간 할당에 대한 노하우가 생긴 것 같다.”

“Streamlit을 이번에 처음 활용해봤는데 생각한 대로 화면이 구성되지 않아 아쉬웠다. 직접 라벨링한 데이터가 모델 학습에 사용된 그 결과가 잘 나와서 성취감이 있었다. 딥러닝 시 자잘한 오류가 많았는데 해결하는 방법을 몰라 애먹었지만, 이번 기회로 인해 다음에 작업할 때 더 수월하게 진행될 것 같다.”

“Streamlit을 활용할 때 파일 구조 고안하는 것이 어려웠다. Yolov5를 그냥 적용시키면 될 줄 알았는데 생각보다 조건이 많이 붙어서 어려움이 많았다. 그럼에도 결과가 좋게 나왔으니 만족한다.”

Q & A

발표를 마치겠습니다.