

Object detection 실습 예제

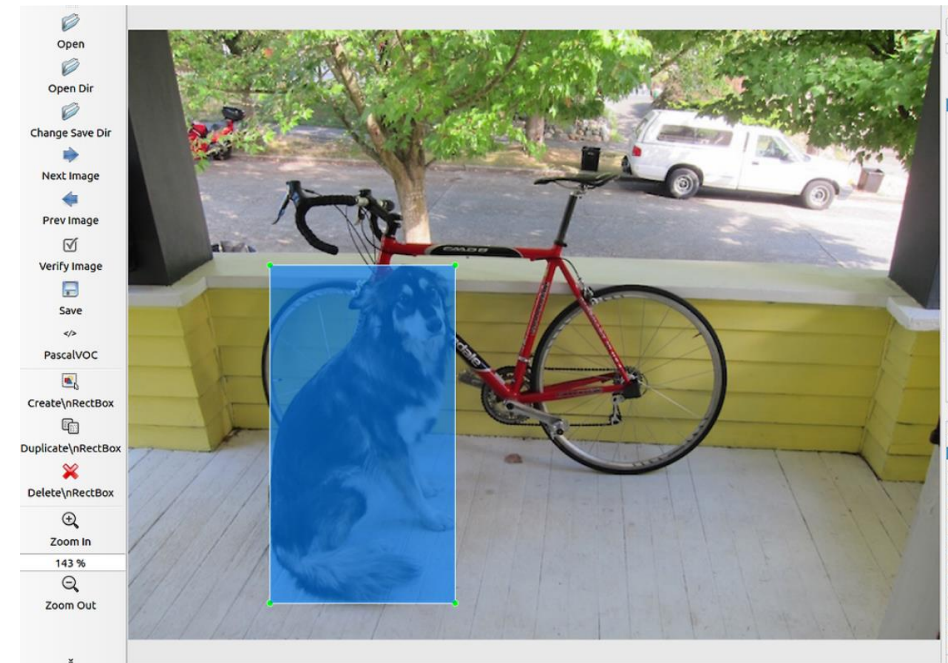
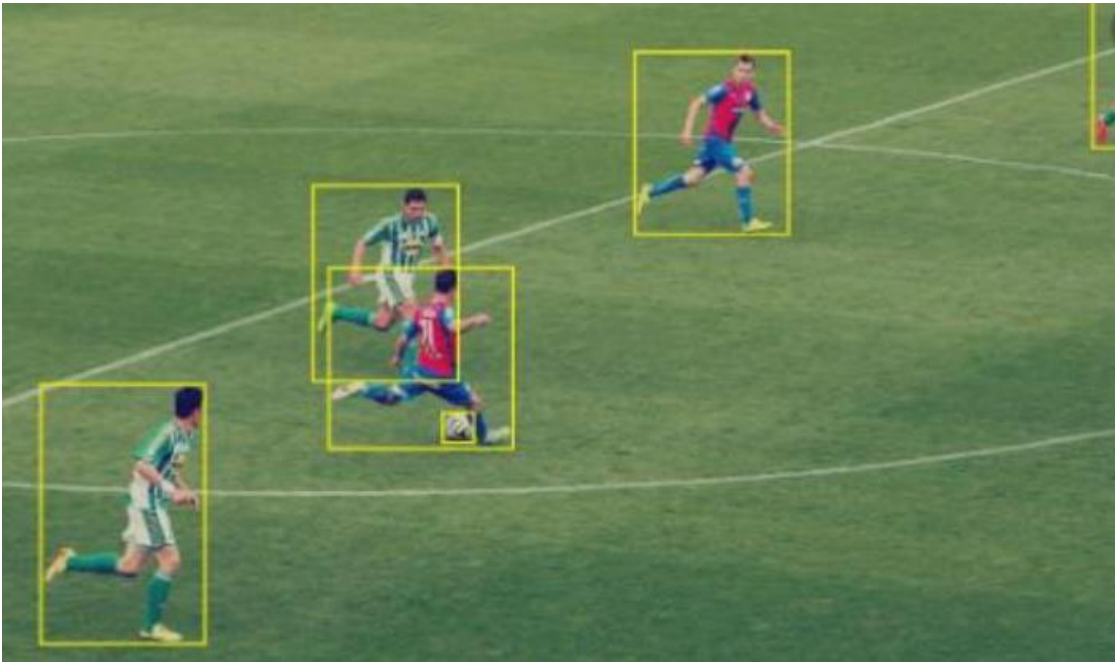
박석희 교수님

실습조교 강정훈
kjh95k@naver.com

Object detection – YOLOv5

Object detection 모델은 지도학습으로 **라벨링된 데이터**를 모델에 학습시키는 과정이 필요함

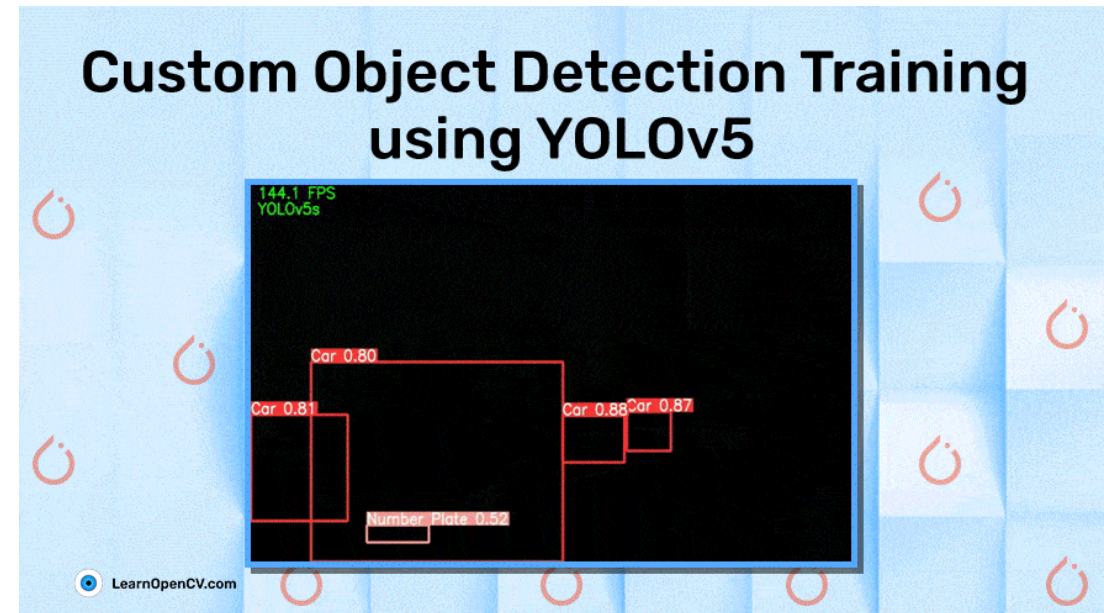
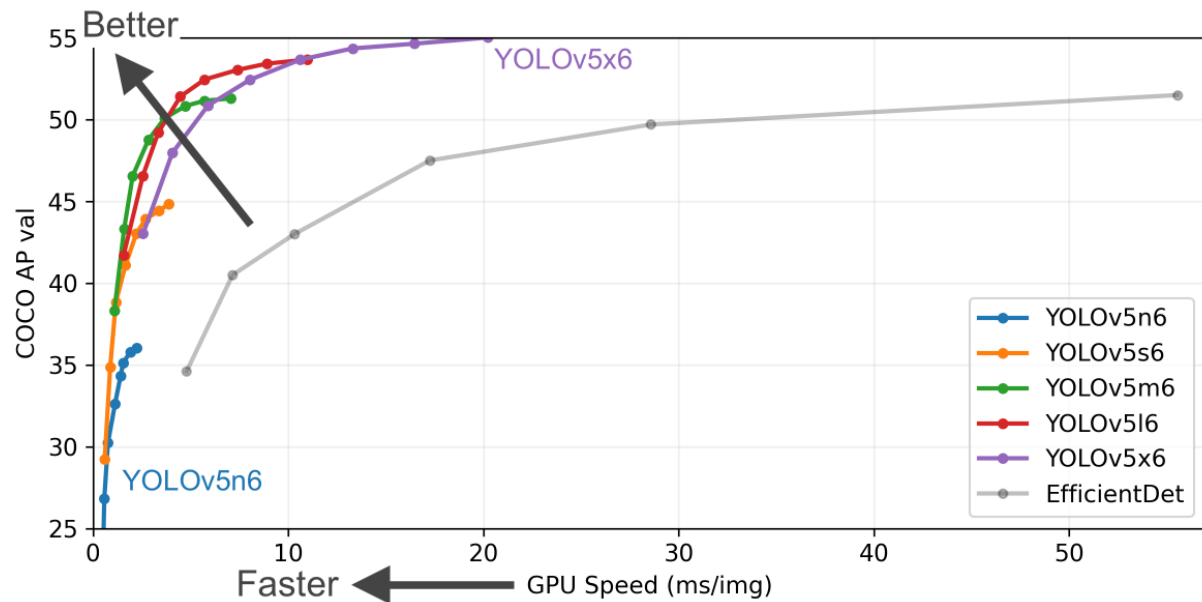
객체를 박스를 이용해 표시하고, 그 객체의 종류를 설정하는 과정을 거쳐야 함.



Object detection – YOLOv5

Object detection 모델 중 YOLOv5를 이용하여 라벨링 및 이미지 내의 객체 탐지 수행

YOLOv5에는 모델 성능에 따라 n,s,m,l,x가 있고, 그 중 YOLOv5s를 이용하여 실습



데이터 다운로드 - 예제1

❖ 객체 탐지 모델 라벨링 데이터 제작 및 탐지 실습 예제

Github 주소 - https://github.com/JeonghunKang1/MMMLab_jh

MMMLab_jh Public

main 1 Branch 0 Tags

JeonghunKang1 ok

File	Status
test	ok
train	ok
valid	ok

README

Add a README

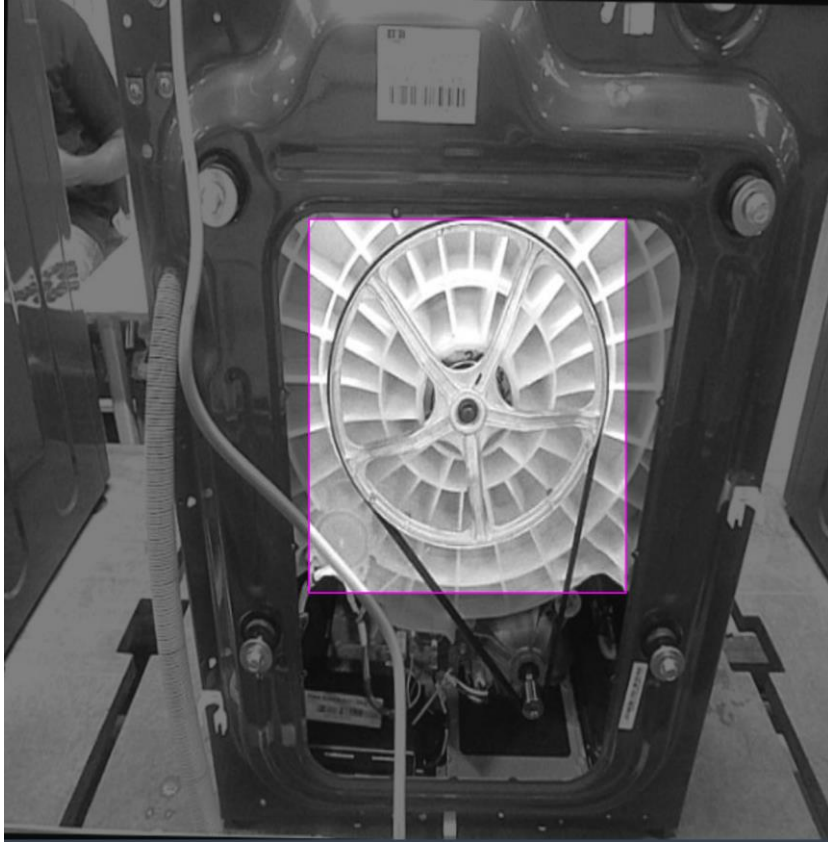
Help people interested in this repository understand your project by adding a README.

Code

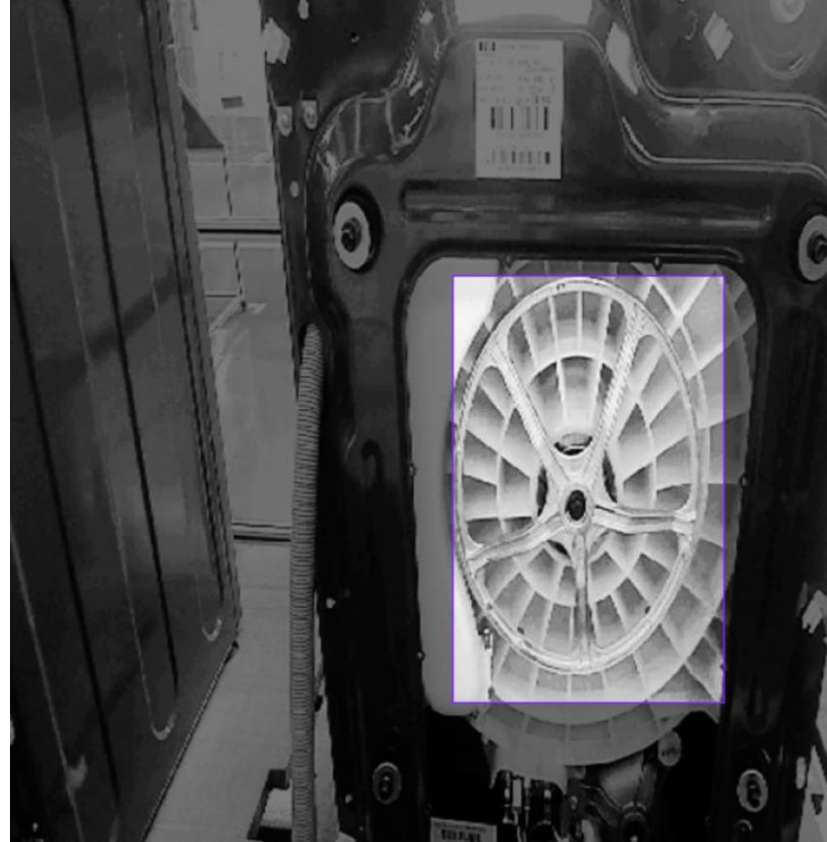
- Local
- Codespaces
- Clone
- HTTPS SSH GitHub CLI
- https://github.com/JeonghunKang1/MMMLab_jh.git
- Clone using the web URL.
- Open with GitHub Desktop
- Download ZIP**

데이터 다운로드 - 예제1

❖ 세탁기 pulley 체결 유무 detection 예제



체결 됨



체결 안됨

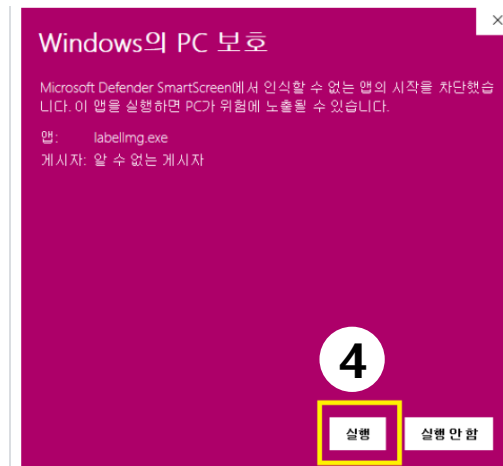
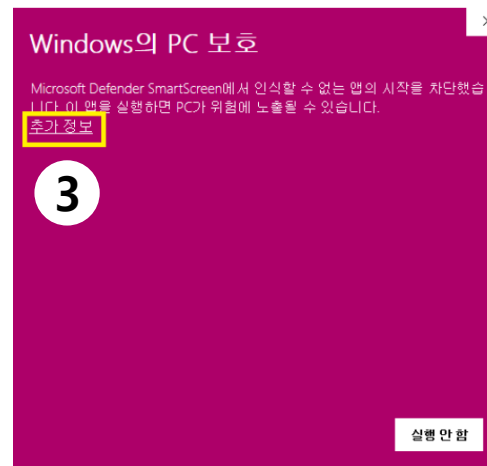
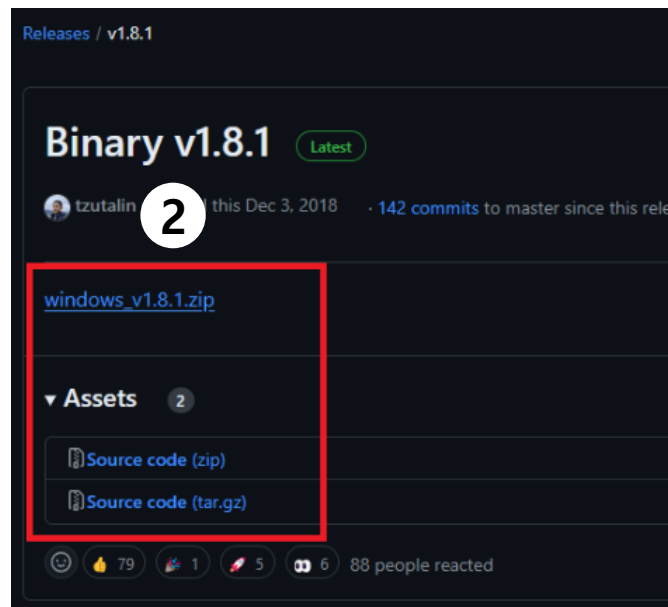
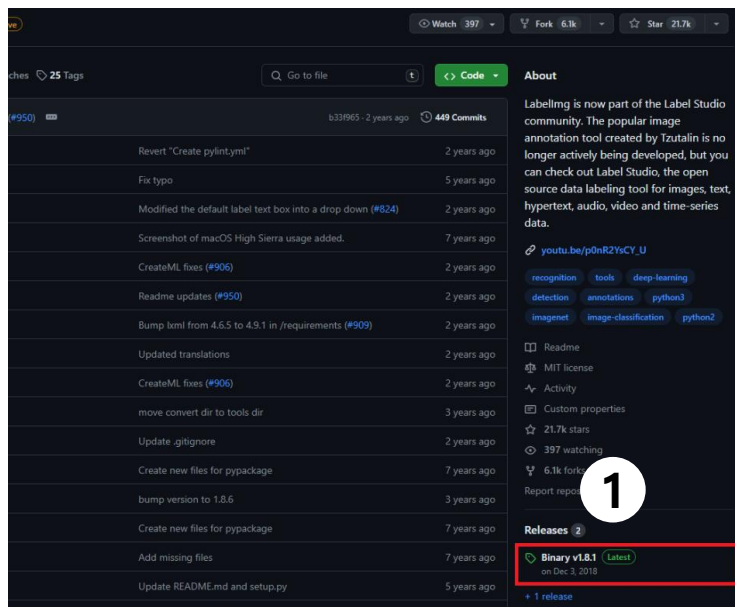
- 데이터는 세탁기 후면 사진으로, pulley 체결 유무에 대해 라벨링한 데이터가 포함됨

라벨링 작업 – 예제 1

❖ labellmg tool을 이용하여 test에 사용할 데이터 셋 생성

→ 대표적인 라벨링 툴인 labellmg를 이용하여 라벨링

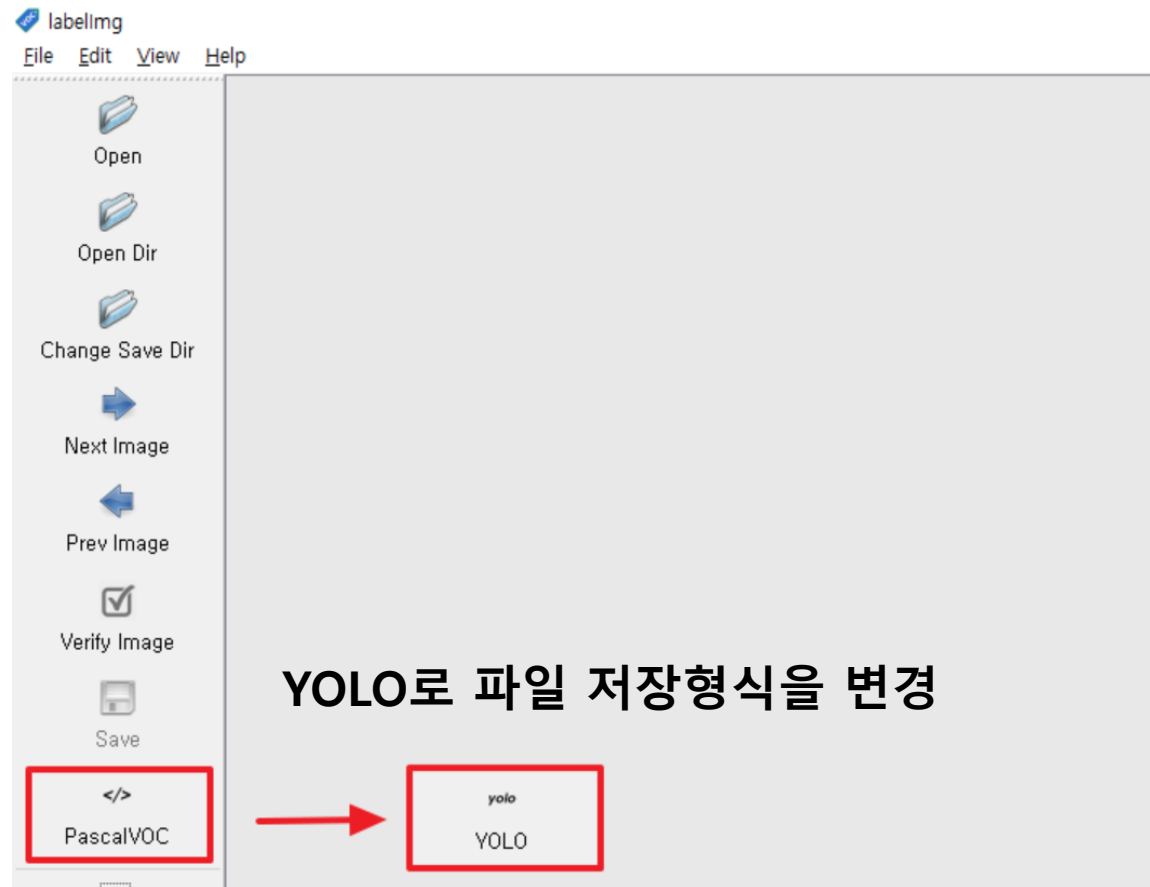
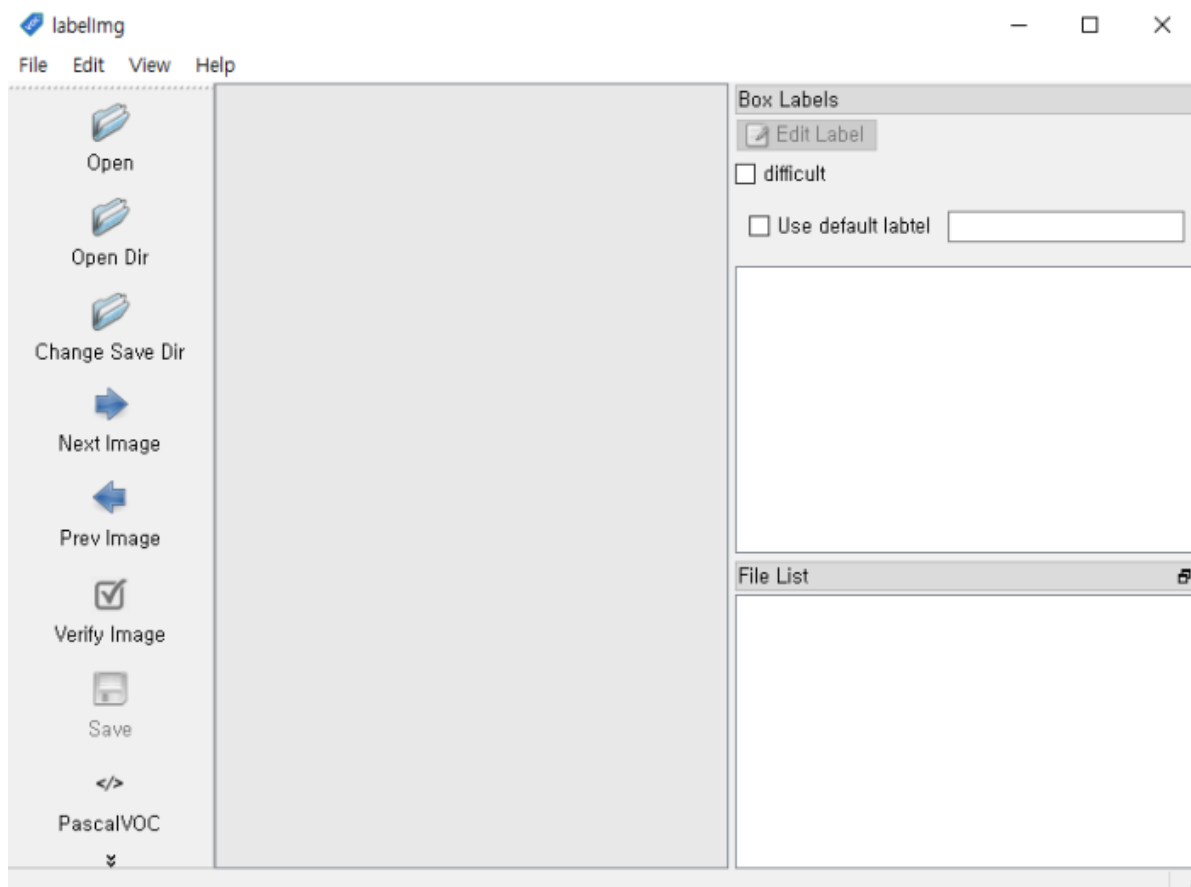
<https://github.com/tzutalin/labellmg>



1. Github에서 windows_v1.8.1.zip 파일 설치

2. C 또는 D 드라이브 바로 아래에서 압축 해제하고 labellmg.exe 실행

라벨링 작업 – 예제 1



(※주의) 검은 화면 종료하면 함께 종료됨

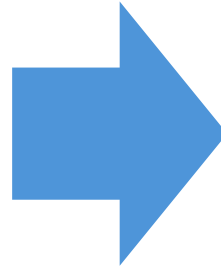
라벨링 작업 – 예제 1

predefined_classes.txt - Windows 메모장

파일(F) 편집(E) 서식(O) 보기(V) 도움말(H)

dog
person
cat
tv
car
meatballs
marinara sauce
tomato soup
chicken noodle soup
french onion soup
chicken breast
ribs
pulled pork
hamburger
cavity

기본 서식



predefined_classes.txt - Windows 메모장

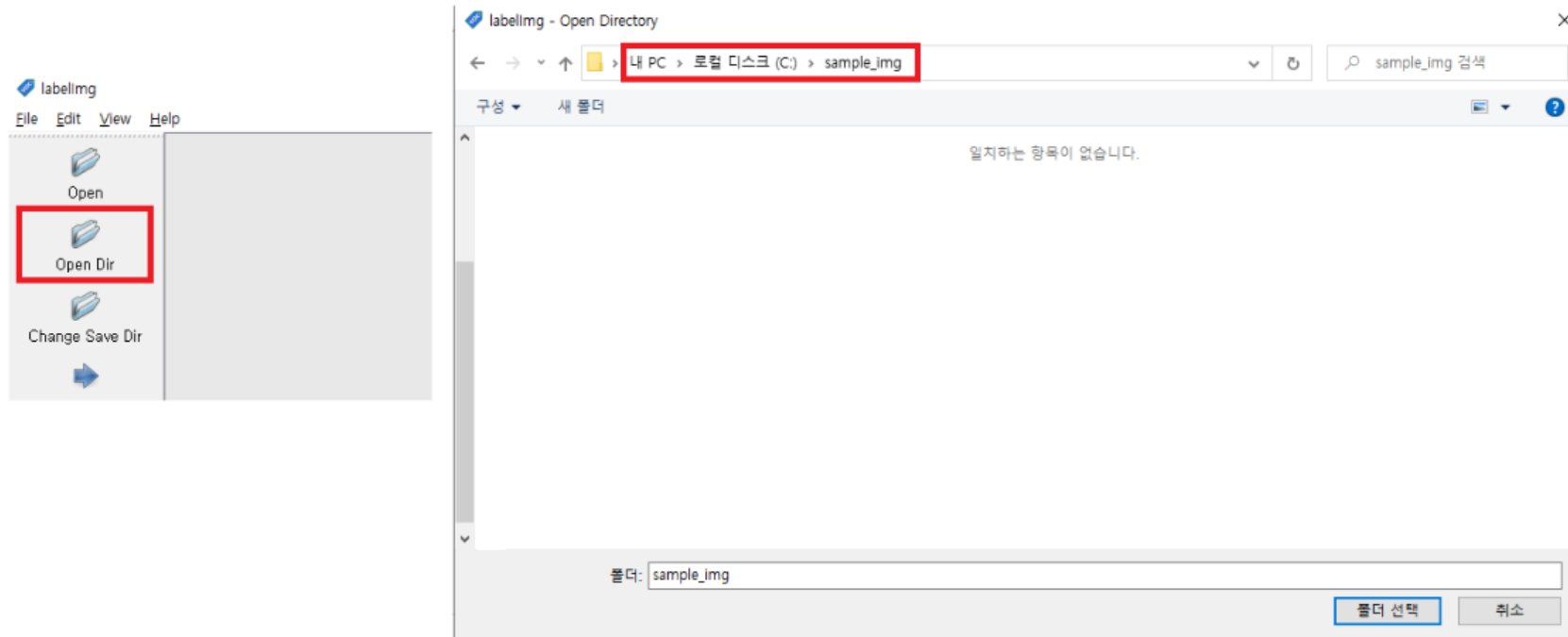
파일(F) 편집(E) 서식(O) 보기(V) 도움말(H)

belt detected
belt not detected

custom 서식

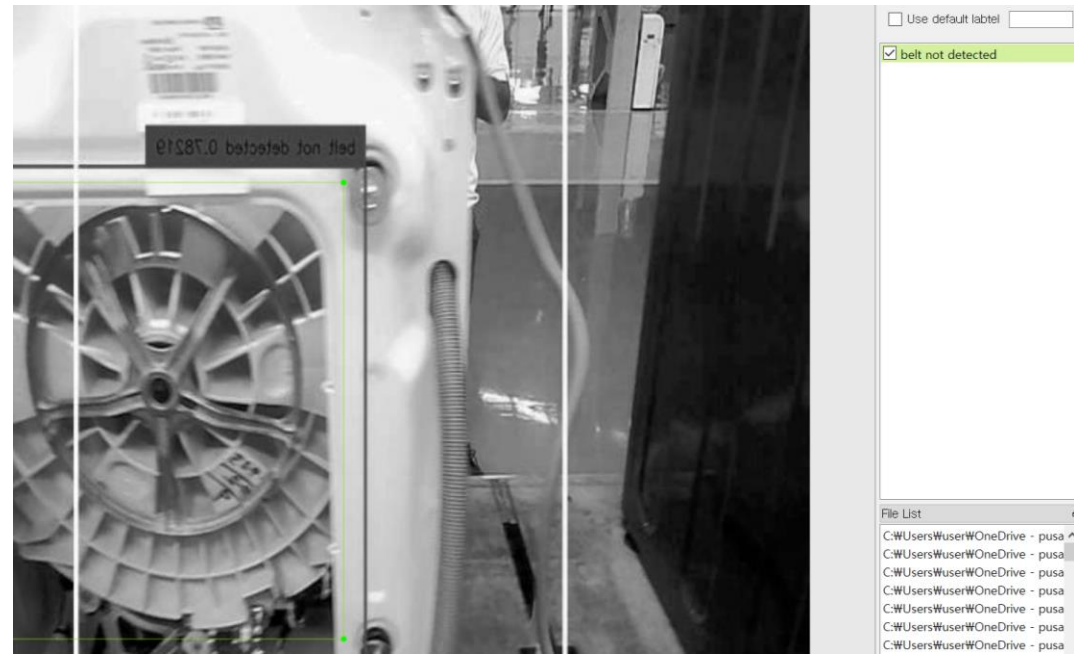
- Predefined_classes를 사용할 label 이름으로 변경한 뒤 저장
- belt detected, belt not detected로 변경

라벨링 작업 – 예제 1



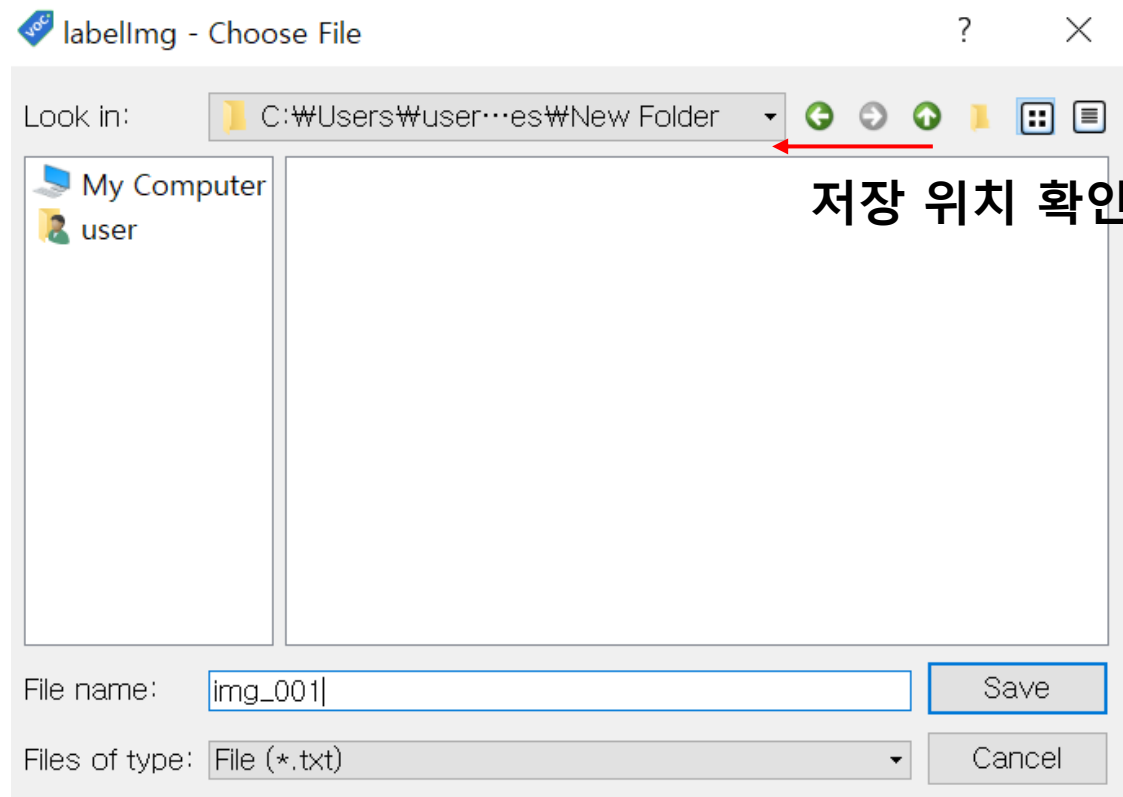
- Open Dir을 클릭해 test/images 폴더를 선택

라벨링 작업 – 예제 1



- W를 눌러 Bounding box를 그려 해당 객체를 라벨링
각 객체에 대해 각각 라벨링 수행

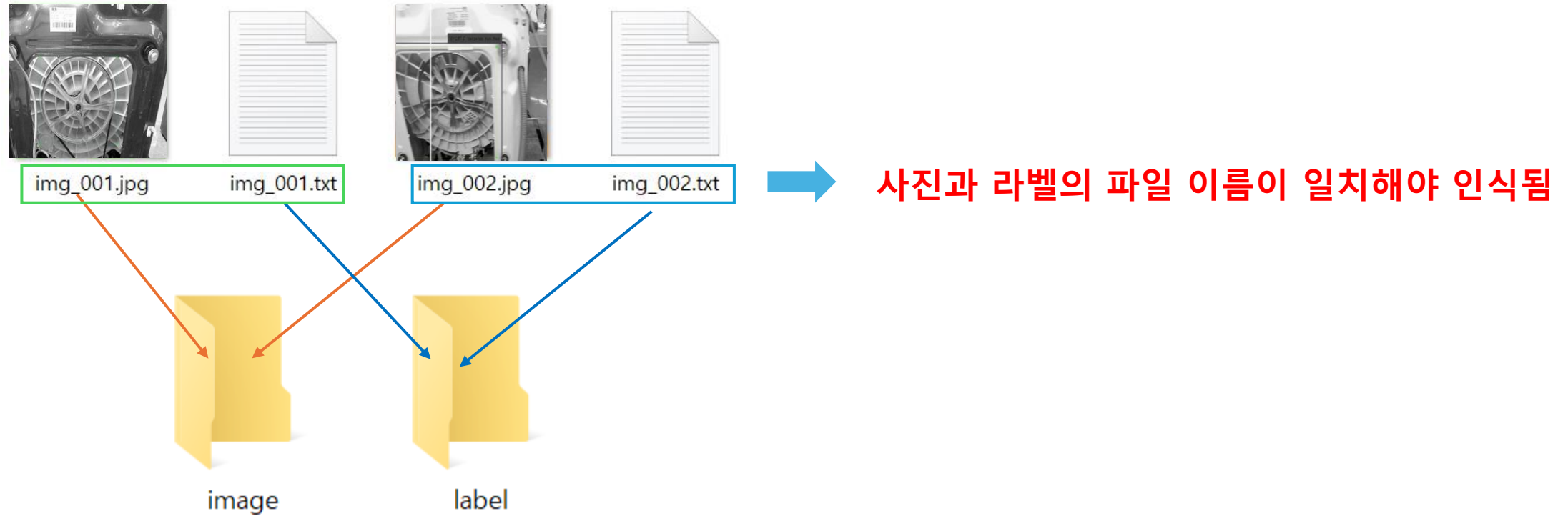
라벨링 작업 – 예제 1



저장 위치 확인 후 저장

작업한 데이터는 사진과 똑같은 이름으로 저장. 제공된 데이터에 대해 라벨링 작업 수행

라벨링 작업 – 예제 1



사진은 test 폴더 내에 image 폴더와 label 폴더에 각각 사진 파일과 labeling 파일 (txt)을 저장

객체 탐지 수행 – 예제 1



data.yaml

Yaml 파일 – 학습 및 평가에 사용할 이미지 데이터의 주소를 포함

```
1  train: ../train/images
2  val: ../valid/images
3  test: ../test/images
4
5  nc: 2
6  names: ['belt detected', 'belt not detected']
```

해당 데이터의 train, val, test의 위치를 설정해주어야 학습 및 detection이 가능

Yaml 파일에서 labeling 이름과 labeling 목록 수를 설정

객체 탐지 수행

```
Miniforge Prompt
(base) C:\Users\USER>mamba activate LGch
(LGch) C:\Users\USER>
```

가상환경이 있다면 실행

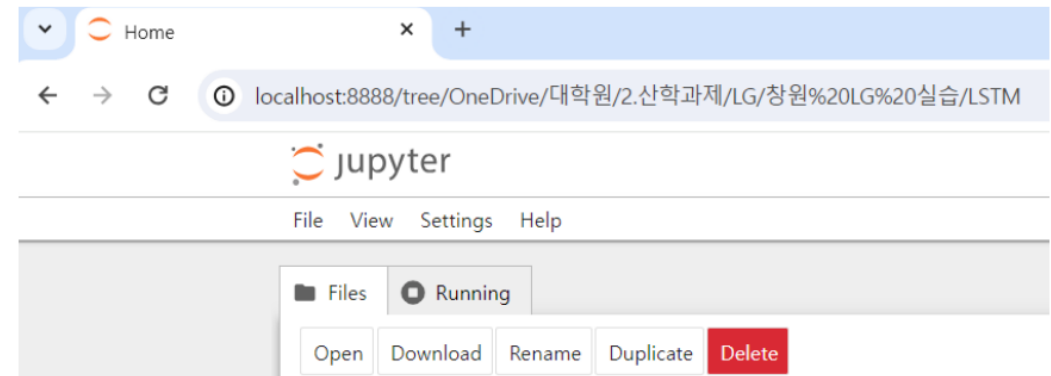
```
(base) C:\Users\USER>mamba create -n envname python=3.9.12
```

가상환경 만들기 코드
envname에 원하는 이름 입력

Jupyter notebook 실행

```
(LGch) C:\Users\USER>jupyter notebook|
```

→ 안 될 경우 pip install jupyter 후 실행



초기 설정

라이브러리 설치

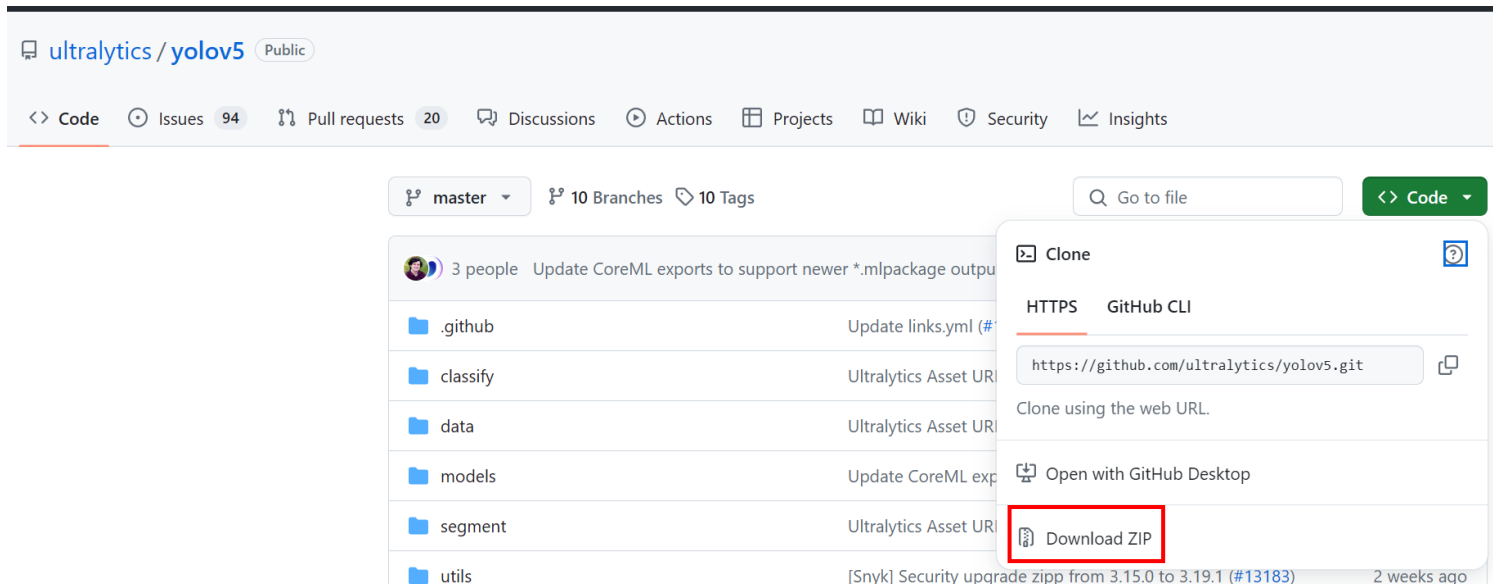
```
!pip install torch
!pip install yaml
!pip install opencv
!pip install matplotlib
!pip install tqdm
!pip install pandas
!pip install numpy
```

라이브러리 import

```
import torch
import yaml
from IPython.display import Image, clear_output
import cv2
import matplotlib.pyplot as plt
import os
import glob
import tqdm
import pandas
import git
import numpy
import git
```

초기 설정

<https://github.com/ultralytics/yolov5>



git hub에서 yolov5 모델 가져오기

=> 설치된 yolov5 폴더 경로 확인

초기 설정

git 설치 및 path 추가

<https://git-scm.com/download/win>

Download for Windows

Click here to download the latest (2.45.2) 64-bit version of Git for Windows. This is the most recent maintained build. It was released about 2 months ago, on 2024-06-03.

Other Git for Windows downloads

Standalone Installer

32-bit Git for Windows Setup.

64-bit Git for Windows Setup.

Portable ("thumbdrive edition")

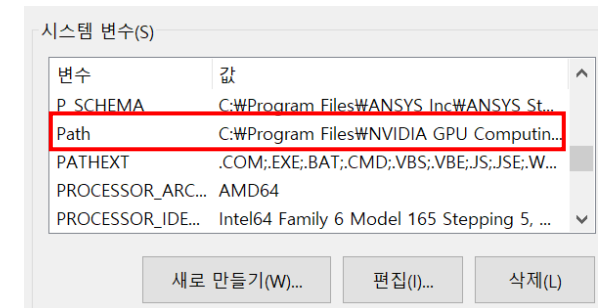
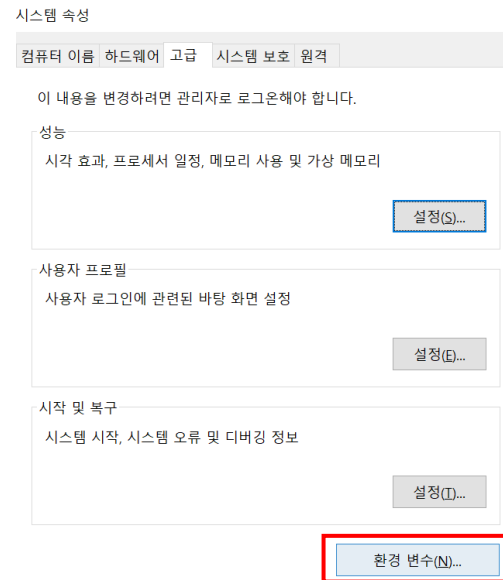
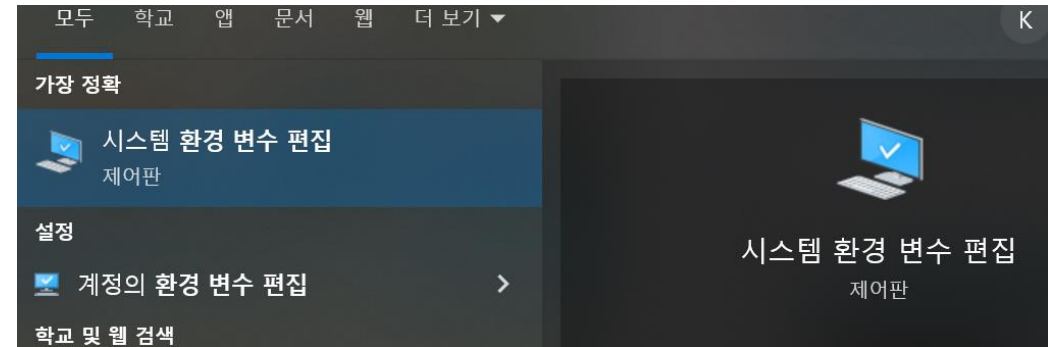
32-bit Git for Windows Portable.

64-bit Git for Windows Portable.

Using winget tool

Install winget tool if you don't already have it, then type this command in command prompt or Powershell.

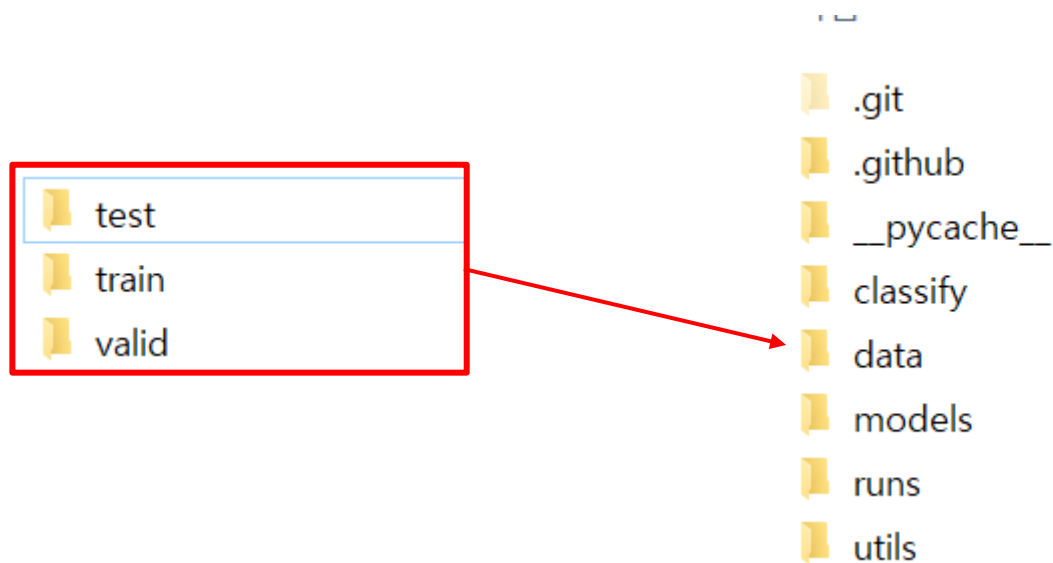
설치 후 exe 실행해서 다 ok 눌러서 진행



Path 접속 후 주소 추가

C:\Program Files\Git\cmd
C:\Program Files\Git\bin\git.exe → (Git이 설치된 주소)

초기 설정



설치된 yolov5/data 폴더로 train, valid, test 폴더 옮기기

초기 설정

학습 및 평가 데이터 목록 파일 생성

```
# 이미지들의 주소 리스트로 만들어줌
train_img_list = glob('C:/Users/USER/kang/yolov5/data/train/images/*.jpg')
val_img_list = glob('C:/Users/USER/kang/yolov5/data/val/images/*.jpg')

# 리스트를 txt파일로 생성
with open('C:/Users/USER/kang/yolov5/data/train.txt', 'w') as f:
    f.write('\n'.join(train_img_list) + '\n')

with open('C:/Users/USER/kang/yolov5/data/val.txt', 'w') as f:
    f.write('\n'.join(val_img_list) + '\n')
```

Ex)

```
./data/train/images#00d0c609-55.jpg
./data/train/images#00d3e705-65.jpg
./data/train/images#00ed7adc-10.jpg
./data/train/images#0133f990-56.jpg
./data/train/images#01994431-60.jpg
./data/train/images#01ae641e-2.jpg
./data/train/images#01e94c89-39.jpg
./data/train/images#0267b139-52.jpg
⋮
```

train.txt, valid.txt, test.txt 파일 경로 들어간 파일을 data 폴더 내에 생성

초기 설정

data.yaml 파일 생성

```
1  train: ../train/images
2  valid: ../valid/images
3  test: ../test/images
4
5  nc: 2
6  names: ['belt detected', 'belt not detected']
```

→ 생성한 .txt 주소로 변경

data.yaml 파일을 yolov5/data 폴더로 옮기고 내용을
탐지하려는 데이터의 주소 및 라벨명으로 변경

데이터셋 위치 정보와 라벨링 정보, class 수 및 class 명이 포함된 yaml 파일 생성 및 수정

초기 설정

학습 파라미터 확인 및 조절

```
431
432 def parse_opt(known=False):
433     parser = argparse.ArgumentParser()
434     parser.add_argument('--weights', type=str, default=ROOT / 'yolov5s.pt', help='initial weights path')
435     parser.add_argument('--cfg', type=str, default='', help='model.yaml path')
436     parser.add_argument('--data', type=str, default=ROOT / 'data/coco128.yaml', help='dataset.yaml path')
437     parser.add_argument('--hyp', type=str, default=ROOT / 'data/hyps/hyp.scratch-low.yaml', help='hyperparameters path')
438     parser.add_argument('--epochs', type=int, default=300, help='total training epochs')
439     parser.add_argument('--batch-size', type=int, default=16, help='total batch size for all GPUs, -1 for autobatch')
440     parser.add_argument('--imgsz', '--img', '--img-size', type=int, default=640, help='train, val image size (pixels)')
441     parser.add_argument('--rect', action='store_true', help='rectangular training')
442     parser.add_argument('--resume', nargs='?', const=True, default=False, help='resume most recent training')
443     parser.add_argument('--nosave', action='store_true', help='only save final checkpoint')
444     parser.add_argument('--noval', action='store_true', help='only validate final epoch')
445     parser.add_argument('--noautoanchor', action='store_true', help='disable AutoAnchor')
446     parser.add_argument('--noplots', action='store_true', help='save no plot files')
447     parser.add_argument('--evolve', type=int, nargs='?', const=300, help='evolve hyperparameters for x generations')
448     parser.add_argument('--bucket', type=str, default='', help='gsutil bucket')
449     parser.add_argument('--cache', type=str, nargs='?', const='ram', help='--cache images in "ram" (default) or "disk"')
450     parser.add_argument('--image-weights', action='store_true', help='use weighted image selection for training')
451     parser.add_argument('--device', default='', help='cuda device, i.e. 0 or 0,1,2,3 or cpu')
452     parser.add_argument('--multi-scale', action='store_true', help='vary img-size +/- 50%%')
453     parser.add_argument('--single-cls', action='store_true', help='train multi-class data as single-class')
454     parser.add_argument('--optimizer', type=str, choices=['SGD', 'Adam', 'AdamW'], default='SGD', help='optimizer')
455     parser.add_argument('--sync-bn', action='store_true', help='use SyncBatchNorm, only available in DDP mode')
456     parser.add_argument('--workers', type=int, default=8, help='max dataloader workers (per RANK in DDP mode)')
457     parser.add_argument('--project', default=ROOT / 'runs/train', help='save to project/name')
458     parser.add_argument('--name', default='exp', help='save to project/name')
459     parser.add_argument('--exist-ok', action='store_true', help='existing project/name ok, do not increment')
460     parser.add_argument('--quad', action='store_true', help='quad dataloader')
461     parser.add_argument('--cos-lr', action='store_true', help='cosine LR scheduler')
462     parser.add_argument('--label-smoothing', type=float, default=0.0, help='Label smoothing epsilon')
463     parser.add_argument('--patience', type=int, default=100, help='EarlyStopping patience (epochs without improvement)')
464     parser.add_argument('--freeze', nargs='+', type=int, default=[0], help='Freeze layers: backbone=10, first3=0 1 2')
465     parser.add_argument('--save-period', type=int, default=-1, help='Save checkpoint every x epochs (disabled if < 1)')
466     parser.add_argument('--seed', type=int, default=0, help='Global training seed')
467     parser.add_argument('--local_rank', type=int, default=-1, help='Automatic DDP Multi-GPU argument, do not modify')
468
```

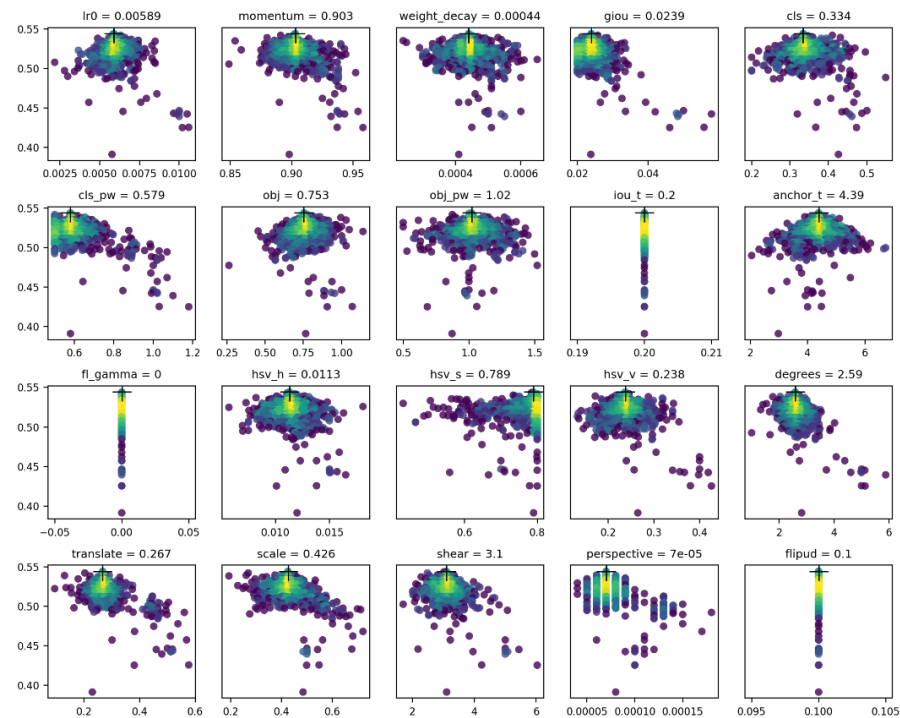
train.py 파일 내에서 학습과 관련된 파라미터의 기본값을 확인 및 변경 가능
→ 설정 안할 시 train.py 파일의 기본값으로 실행됨

초기 설정

부가 기능) 모델의 하이퍼 파라미터 자동 설정 기능 – evolution

```
# Evolve hyperparameters (optional)
else:
    # Hyperparameter evolution metadata (mutation scale 0-1, lower_limit, upper_limit)
    meta = {
        'lr0': (1, 1e-5, 1e-1), # initial learning rate (SGD=1E-2, Adam=1E-3)
        'lrf': (1, 0.01, 1.0), # final OneCycleLR learning rate (lr0 * lrf)
        'momentum': (0.3, 0.6, 0.98), # SGD momentum/Adam beta1
        'weight_decay': (1, 0.0, 0.001), # optimizer weight decay
        'warmup_epochs': (1, 0.0, 5.0), # warmup epochs (fractions ok)
        'warmup_momentum': (1, 0.0, 0.95), # warmup initial momentum
        'warmup_bias_lr': (1, 0.0, 0.2), # warmup initial bias lr
        'box': (1, 0.02, 0.2), # box loss gain
        'cls': (1, 0.2, 4.0), # cls loss gain
        'cls_pw': (1, 0.5, 2.0), # cls BCELoss positive_weight
        'obj': (1, 0.2, 4.0), # obj loss gain (scale with pixels)
        'obj_pw': (1, 0.5, 2.0), # obj BCELoss positive_weight
        'iou_t': (0, 0.1, 0.7), # IoU training threshold
        'anchor_t': (1, 2.0, 8.0), # anchor-multiple threshold
        'anchors': (2, 2.0, 10.0), # anchors per output grid (0 to ignore)
        'fl_gamma': (0, 0.0, 2.0), # focal loss gamma (efficientDet default gamma=1.5)
        'hsv_h': (1, 0.0, 0.1), # image HSV-Hue augmentation (fraction)
        'hsv_s': (1, 0.0, 0.9), # image HSV-Saturation augmentation (fraction)
        'hsv_v': (1, 0.0, 0.9), # image HSV-Value augmentation (fraction)
        'degrees': (1, 0.0, 45.0), # image rotation (+/- deg)
        'translate': (1, 0.0, 0.9), # image translation (+/- fraction)
        'scale': (1, 0.0, 0.9), # image scale (+/- gain)
        'shear': (1, 0.0, 10.0), # image shear (+/- deg)
        'perspective': (0, 0.0, 0.001), # image perspective (+/- fraction), range 0-0.001
        'flipud': (1, 0.0, 1.0), # image flip up-down (probability)
        'fliplr': (0, 0.0, 1.0), # image flip left-right (probability)
        'mosaic': (1, 0.0, 1.0), # image mixup (probability)
        'mixup': (1, 0.0, 1.0), # image mixup (probability)
        'copy_paste': (1, 0.0, 1.0)} # segment copy-paste (probability)
```

python train.py --weights yolov5s.pt --evolve



모델에 적절한 hyperparameter를 유전 알고리즘을 이용해 자동으로 내 데이터 셋에 적합하게 설정하는 기능

그 때 탐색할 hyperparameter 범위를 직접 설정할 수 있음

초기 설정

부가 기능) 이미지 비율이 정사각형이 아닌 경우

```
python train.py ..... --weights yolov5s.pt --rect
```

기본적으로 학습 이미지는 정사각형으로 학습되고, 직사각형 이미지의 경우 정사각형 비율로 일부 왜곡되어 학습됨.

--rect를 사용할 경우 직사각형 이미지의 가로 세로 중 긴 부분에 맞춰서 직사각형으로 학습됨

→ ex) --img 1080 --rect => (1080x720)으로 학습됨

객체 탐지 수행

객체 탐지 수행

예시 문구)

`!python train.py --img 640 --epochs 3 --data coco128.yaml --weights yolov5s.pt`

The diagram illustrates the command line arguments for the training script. A red arrow points from the text '앞에 주소 넣어야 함' (Must enter address at the front) to the exclamation mark at the start of the command. Blue arrows point from descriptive text to specific arguments: '전체 데이터 학습 반복 수' (Total number of learning iterations for all data) points to '--epochs 3'; '학습할 때의 이미지 크기' (Image size when learning) points to '--img 640'; '사용할 데이터 정보 (data.yaml 이용)' (Data information to be used (using data.yaml)) points to '--data coco128.yaml'; and '학습에 사용할 weights' (Weights to be used for learning) points to '--weights yolov5s.pt'.

Ex) `!python ~/train.py --img 320 --batch 5 --epochs 3 --device cpu --data ~/data.yaml --weights ~/yolov5s.pt`

train.py 파일을 실행시켜 yolov5s.pt를 통해 학습 수행

※ CPU 학습이기 때문에 batch와 epoch은 작게 설정

추가로 사용할 수 있는 요소

--batch => 한 번에 학습하는 이미지(데이터) 수 → 학습 정확도와 관련

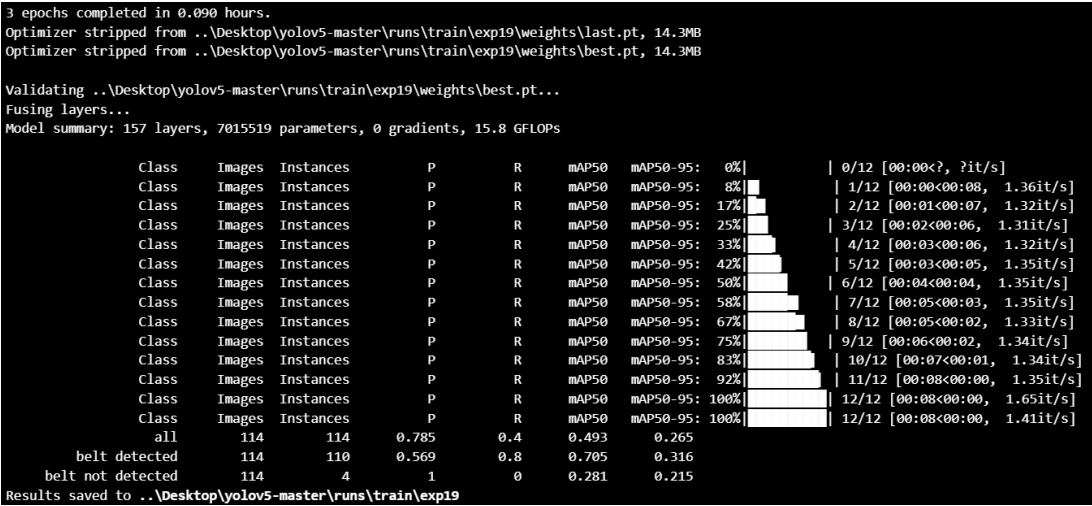
--cache => 학습 데이터 기록 → 학습 과정 확인을 위해 사용

⋮

객체 탐지 수행

객체 탐지 수행

CUDA 설치가 되어 있지 않아 CPU로 탐지 수행 -> 속도 제약 및 가능한 이미지 크기 한계 있음



높은 성능의 GPU 사용 시 더 큰 크기의 이미지 학습
및 한 번에 학습 가능한 데이터 수를 크게 할 수 있음

* 완료 시에 표시됨

객체 탐지 수행

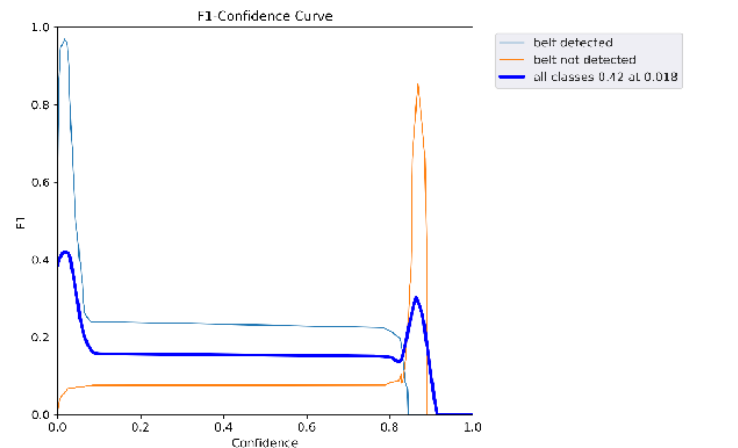
```
# 텐서보드 실행
%load_ext tensorboard
%tensorboard --logdir runs
```

F1_curve



Step 99

● train/exp7



Tensorboard를 실행시키면 학습 과정 및 mAP, F1 score 등 다양한 학습 결과를 확인 가능

객체 탐지 수행

학습 모델 이용 탐지 수행

```
!python detect.py --weights runs/train/exp{train_exp_num}/weights/best.pt --img 416 --conf 0.3 --source {test_data_dir}
```

↓
학습된 모델의 최고 성능의 weights 사용

↑
탐지를 수행할 이미지 폴더

↓
Confidence threshold : 객체를 탐지할 최소 정확도
Ex) 0.3 보다 작은 정확도를 가질 경우 객체가 없는 것으로 판단

```
68
69 @smart_inference_mode()
70 def run(
71     weights=ROOT / "yolov5s.pt", # model path or triton URL
72     source=ROOT / "data/images", # file/dir/URL/glob/screen/0(webcam)
73     data=ROOT / "data/coco128.yaml", # dataset.yaml path
74     imgsz=(640, 640), # inference size (height, width)
75     conf_thres=0.25, # confidence threshold
76     iou_thres=0.45, # NMS IOU threshold
77     max_det=1000, # maximum detections per image
78     device="", # cuda device, i.e. 0 or 0,1,2,3 or cpu
79     view_img=False, # show results
80     save_txt=False, # save results to *.txt
81     save_csv=False, # save results in CSV format
82     save_conf=False, # save confidences in --save-txt labels
83     save_crop=False, # save cropped prediction boxes
84     nosave=False, # do not save images/videos
85     classes=None, # filter by class: --class 0, or --class 0 2 3
86     agnostic_nms=False, # class-agnostic NMS
87     augment=False, # augmented inference
88     visualize=False, # visualize features
89 )
```

Detect.py 파일에서 다양한 하이퍼 파라미터를 조절 가능

Ex)

conf_thres => 설정 안할 경우 기본값 0.25

iou_thres -> 객체로 탐지하기 위한 최소 IoU 값

Max_det -> 한 이미지 내에서 탐지가능한 최대 객체 수

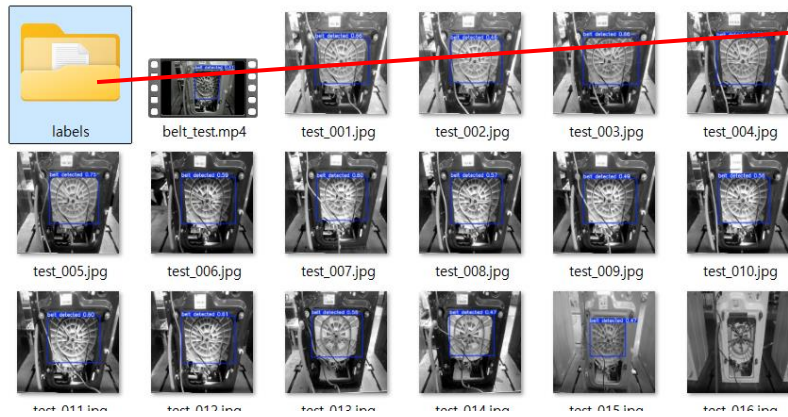
Save_txt -> True로 할 경우 탐지 결과를 txt 파일로 저장 가능

학습된 모델을 이용하여 탐지를 수행하는 코드

객체 탐지 수행

부가 기능 - save-txt

```
1 -conf 0.3 --source C:/Users/USER/Desktop/yolov5-master/data/test/images --save-txt
```



belt_test_1.txt	2024-07-30 오전 10:06	텍스트 문서	1KB
belt_test_2.txt	2024-07-30 오전 10:06	텍스트 문서	1KB
belt_test_3.txt	2024-07-30 오전 10:06	텍스트 문서	1KB
belt_test_4.txt	2024-07-30 오전 10:06	텍스트 문서	1KB
belt_test_5.txt	2024-07-30 오전 10:06	텍스트 문서	1KB
belt_test_6.txt	2024-07-30 오전 10:06	텍스트 문서	1KB
belt_test_7.txt	2024-07-30 오전 10:06	텍스트 문서	1KB
belt_test_8.txt	2024-07-30 오전 10:06	텍스트 문서	1KB
belt_test_9.txt	2024-07-30 오전 10:06	텍스트 문서	1KB
belt_test_10.txt	2024-07-30 오전 10:06	텍스트 문서	1KB
belt_test_11.txt	2024-07-30 오전 10:06	텍스트 문서	1KB
belt_test_12.txt	2024-07-30 오전 10:06	텍스트 문서	1KB
belt_test_13.txt	2024-07-30 오전 10:06	텍스트 문서	1KB
belt_test_14.txt	2024-07-30 오전 10:06	텍스트 문서	1KB
belt_test_15.txt	2024-07-30 오전 10:06	텍스트 문서	1KB
belt_test_16.txt	2024-07-30 오전 10:06	텍스트 문서	1KB
belt_test_17.txt	2024-07-30 오전 10:06	텍스트 문서	1KB

파일 편집 보기

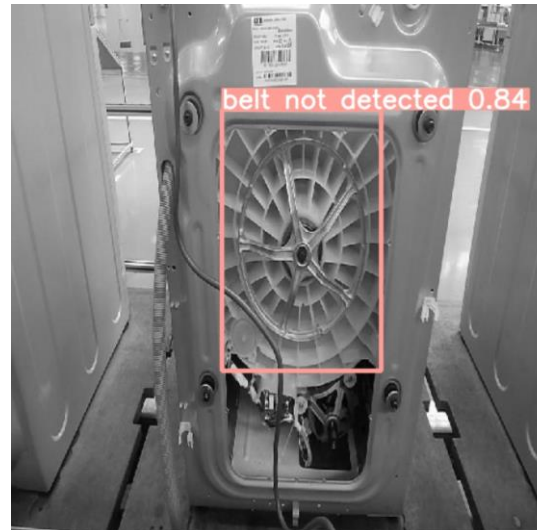
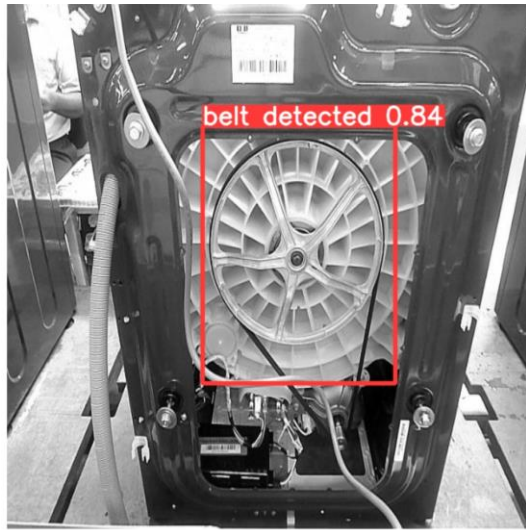
0 0.543911 0.455208 0.223653 0.45625

detect.py 기능을 사용해 탐지를 수행할 때 탐지 결과를 사진과 더불어 숫자 데이터로 얻고 싶을 때 사용

객체 탐지 수행

runs 폴더 내의 detect 폴더에 detect.py를 통해 탐지한 결과가 저장됨

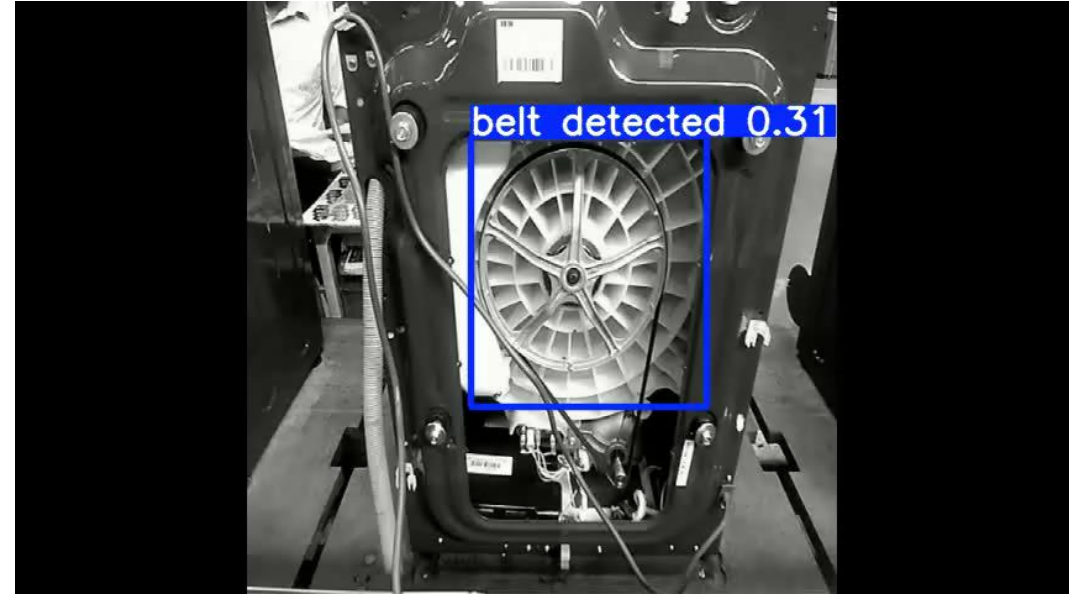
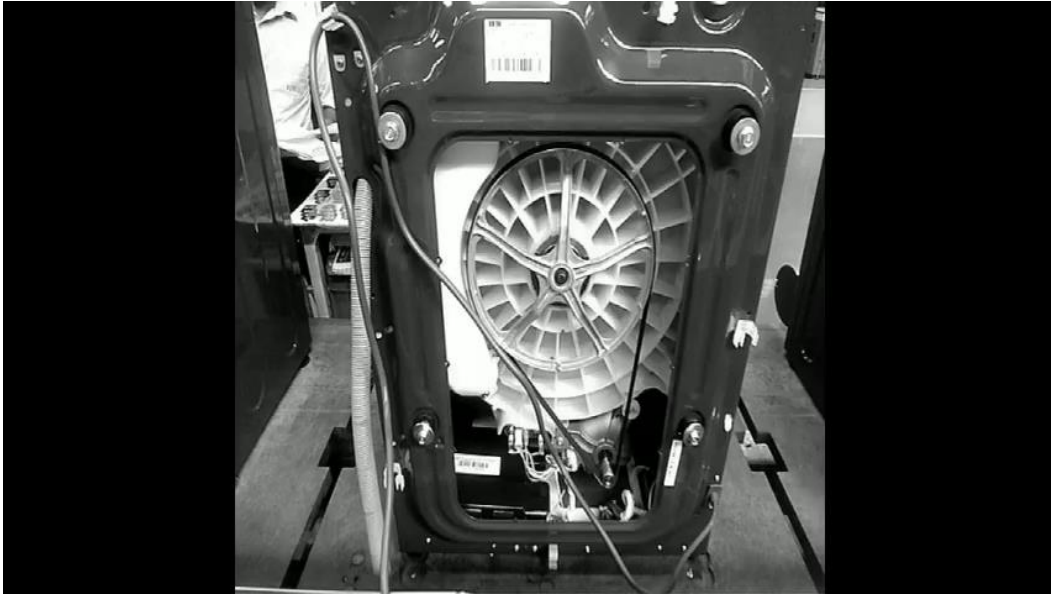
```
for imageName in glob.glob('/content/drive/MyDrive/yolov5/runs/detect/exp' + str(test_exp_num) + '/*.jpg'):  
    display(Image(filename=imageName))  
    print("\n\n")
```



Detect를 수행한 폴더를 불러와 이미지 확인

객체 탐지 수행

영상에 대해 탐지 수행



detect.py 파일을 실행시켜 동일한 코드로 영상에 대해서도 탐지를 수행할 수 있음

객체 탐지 수행

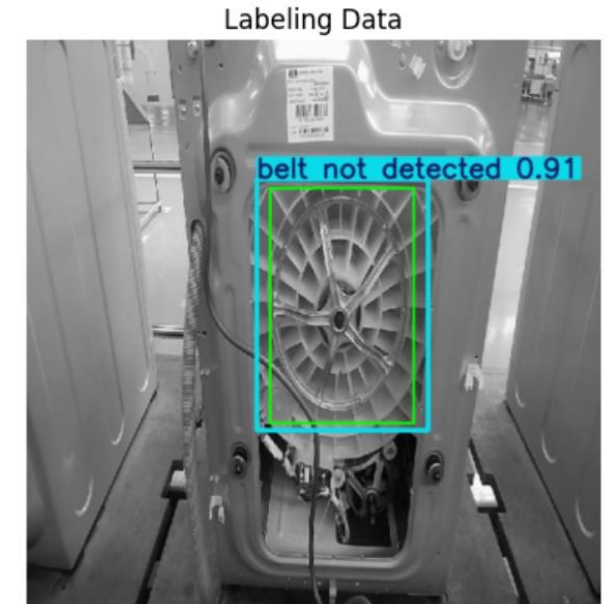
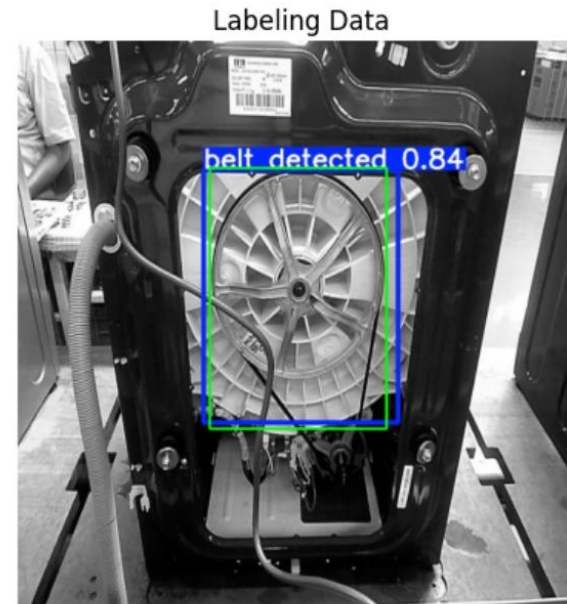
```
import os
import yaml
import cv2
import matplotlib.pyplot as plt
from IPython.display import Image, display
import glob

# 바운딩 박스 그리기 함수
def draw_boxes(image, boxes, color, labels=None):
    for i, box in enumerate(boxes):
        x1, y1, x2, y2 = map(int, box)
        cv2.rectangle(image, (x1, y1), (x2, y2), color, 2)
        if labels:
            cv2.putText(image, labels[i], (x1, y1 - 10), cv2.FONT_HERSHEY_SIMPLEX, 0.9, color, 2)
    return image

def load_labels(label_file, width, height):
    with open(label_file, 'r') as file:
        labels = [list(map(float, line.split())) for line in file.readlines()]
    boxes = [convert_box_format(label, width, height) for label in labels]
    return boxes

def convert_box_format(box, width, height):
    x_center, y_center, w, h = box[1:5]
    x1 = (x_center - w / 2) * width
    y1 = (y_center - h / 2) * height
    x2 = (x_center + w / 2) * width
    y2 = (y_center + h / 2) * height
    return [x1, y1, x2, y2]

# 테스트 결과 시각화
```

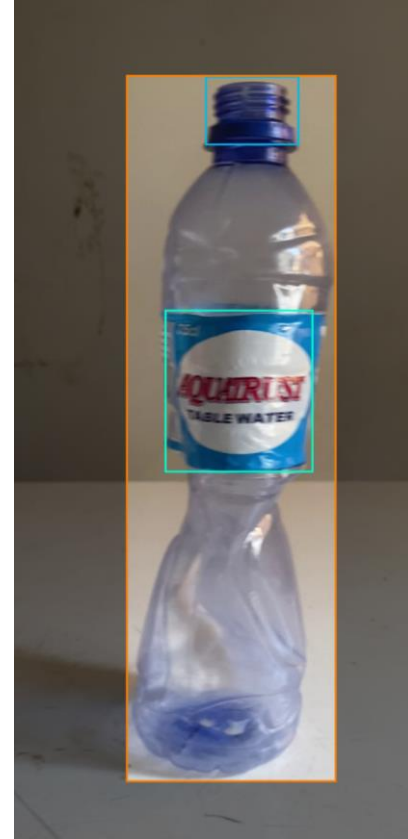
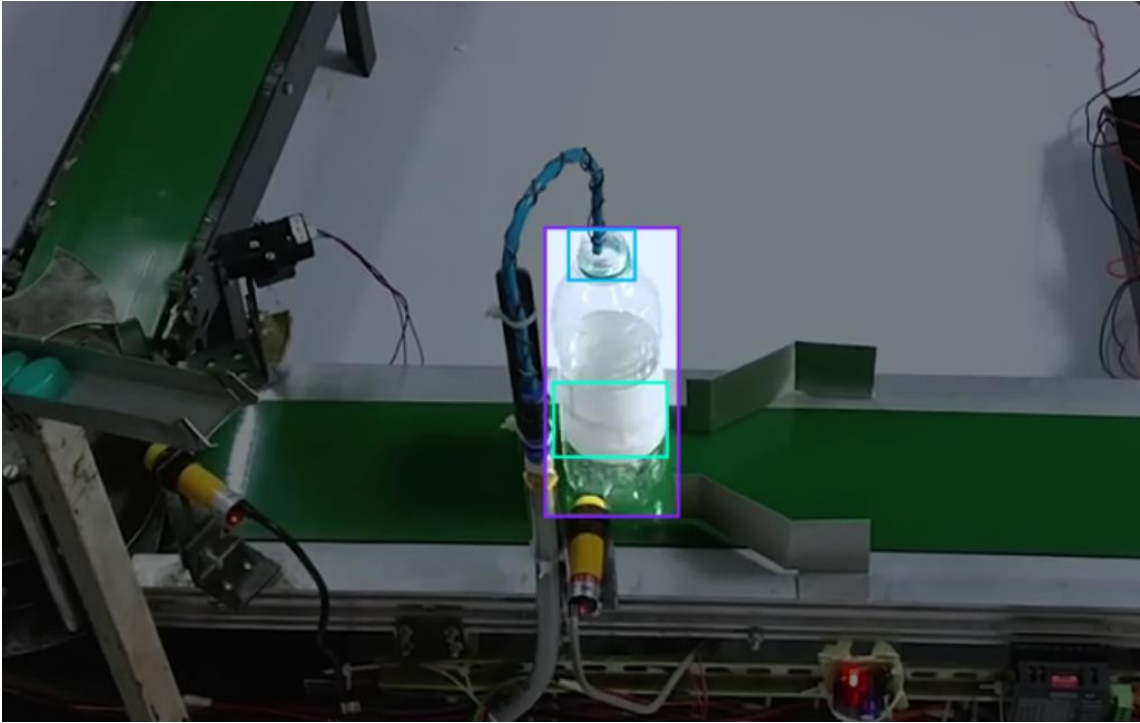


탐지 결과와 labeling 결과 비교

- Detect 결과는 사진에 bounding box가 붙은 채로 출력되기 때문에 탐지 정확도를 수치화해서 출력하는 방법은 아님
detect.py의 save_txt 파라미터를 true로 하면 bounding box의 txt를 추출하여 직접적인 비교에 사용할 수 있음


Multi-class 인 경우 - 예제

한 이미지 내에서 여러 class가 존재하는 경우




Labeling 종류는 4가지로 crumbled, not crumbled, cap, label 4가지에 대해 라벨링 수행

Multi-class 인 경우 - 예제

 predefined_classes.txt - Windows 메모장

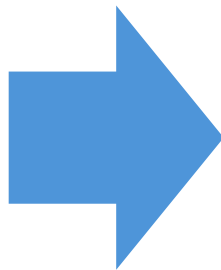
파일(F) 편집(E) 서식(O) 보기(V) 도움말(H)

belt detected
belt not detected

 predefined_classes.txt - Windows 메모장

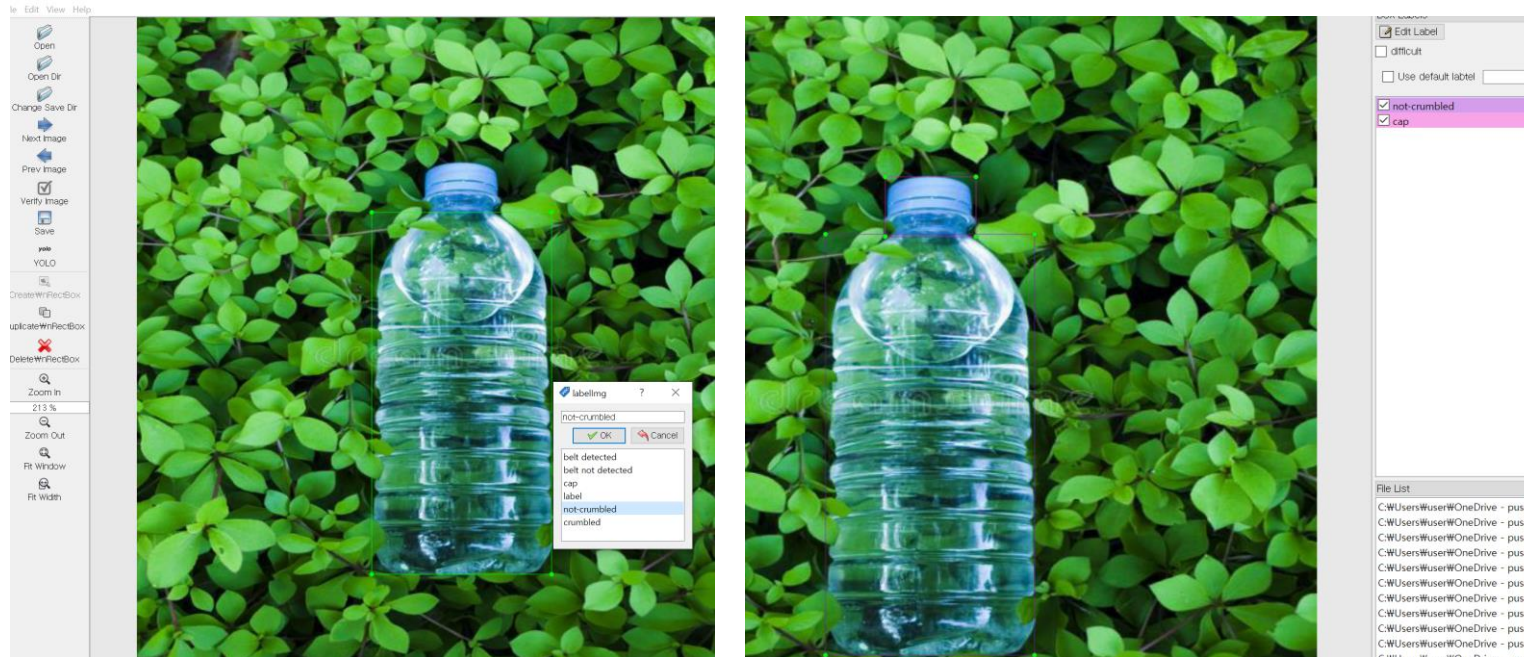
파일(F) 편집(E) 서식(O) 보기(V) 도움말(H)

cap
label
not-crumbled
crumbled



Labeling 종류는 4가지로 crumbled, not crumbled, cap, label 4가지에 대해 라벨링 수행

Multi-class 인 경우 - 예제



```
1 train: ../train/images
2 val: ../valid/images
3 test: ../test/images
4
5 nc: 2 → 4
6 names: ['belt detected', 'belt not detected']
```

↓
['crumbled', 'not crumbled', 'cap', 'label']

감사합니다