INDUSTRIAL AI RESEARCH CENTER

YOLO를 이용한 객체검출 모델 개발

2023. 05. 17

김 현 용

충북대학교 산업인공지능학과

강의 일정

INDUSTRIAL AI RESEARCH CENTER

주차	날짜	발표 주제	비고
1	3/08	**[강의] 오리엔테이션 / 조 편성**	비대면수업
2	3/15	**[강의] Project #1: CNN을 이용한 불량 검출**	대면수업(1)
3	3/22	조별 토의 및 멘토링 , **[강의] OpenCV 기본 명령어**	비대면수업
4	3/29	조별 토의 및 멘토링, **[강의] Numpy와 Matplotlib 시각화**	비대면수업
5	4/05	Project #1 주제발표(5) : 한희주, 명성구, 권진관, 백정흠, 신건철	대면수업(2)
6	4/12	Project #1 주제발표(3) : 박영제, 김현기, 원윤재 , 조별 토의 및 멘토링	비대면수업
7	4/19	프로젝트 최종점검(사전발표)	비대면수업
8	4/26	Project #1 발표평가 (국제회의실 B224)	대면수업(3)
9	5/03	**[학과행사] 가디언별 토의 → 장소, 시간 별도 통보 (22~23학번 통합)**	대면수업(4)
10	5/10	**[강의] Project #2 : YOLO를 이용한 객체 검출 (이론)**	비대면수업
11	5/17	조별 토의 및 멘토링, **[강의] CUDA 및 YOLO 환경구성, YOLO를 이용한 객체 검출 (실제)**	대면수업(5)
12	5/24	Project #2 주제발표(10) : 이선명, 김홍열, 임강혁, 안성인, 송동건, 이재익, 이정현, 장현우, 한병엽, 이진우	비대면수업
13	5/31	조별 토의 및 멘토링	비대면수업
14	6/07	프로젝트 최종점검(사전발표) : 테스트 데이터 공개 → 검출결과 제출	비대면수업
15	6/14	Project #2 발표평가 (국제회의실 B224)	대면수업(6)

CUDA 설치

INDUSTRIAL AI RESEARCH CENTER

- 그래픽 카드 확인 : 장치관리자 > 디스플레이 어댑터
- NVIDIA 드라이버 설치 → https://www.nvidia.com/Download/index.aspx?lang=kr
- 커맨드창 nvidia-smi로 확인
- CUDA Toolkit 설치 → https://developer.nvidia.com/cuda-toolkit-archive

  CUDA Toolkit 11.7.1 (August 2022), **Versioned Online Documentation**

- cuDNN 라이브러리 추가 : 다운로드 후 압축을 풀어서 CUDA 폴더에 복사

NVIDIA cuDNN is a GPU-accelerated library of primitives for deep neural networks.

Download cuDNN v8.8.1 (March 8th, 2023), for CUDA 12.x

Download cuDNN v8.8.1 (March 8th, 2023), for CUDA 11.x

Local Installers for Windows and Linux, Ubuntu(x86_64, armsbsa)

Local Installer for Windows (Zip)

Local Installer for Linux x86_64 (Tar)

Local Installer for Linux PPC (Tar)

Pytorch 설치

INDUSTRIAL AI RESEARCH CENTER

- PyTorch 설치 버전 확인 → https://pytorch.org

PyTorch Build	Stable (2.0.0)		Preview (Nightly)	
Your OS	Linux	Mac	Windows	
Package	Conda	Pip	LibTorch	Source
Language	Python		C++ / Java	
Compute Platform	CUDA 11.7	CUDA 11.8	~~ROCm 5.4.2~~	CPU
Run this Command:	pip3 install torch torchvision torchaudio --index-url https://download.pytorch.org/whl/cu117			

Linux and Windows

```
# ROCM 5.2 (Linux only)
pip3 install torch torchvision torchaudio --extra-index-url
pip install torch==1.13.1+rocm5.2 torchvision==0.14.1+rocm5.2 torchaudio==0.13.1 --extra-index-url https://
# CUDA 11.6
pip install torch==1.13.1+cu116 torchvision==0.14.1+cu116 torchaudio==0.13.1 --extra-index-url https://downl
# CUDA 11.7
pip install torch==1.13.1+cu117 torchvision==0.14.1+cu117 torchaudio==0.13.1 --extra-index-url https://downl
```

PyCharm 설치 (#1/3)

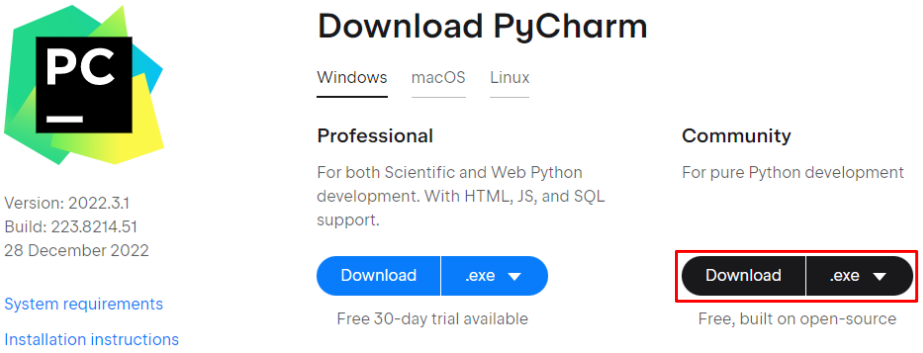
INDUSTRIAL AI RESEARCH CENTER

▣ **통합개발환경 (IDE, integrated development environment)**

- **코드 편집기(editor) + 실행기(compiler/interpreter) + 디버거(debugger)**
- 가독성 : Keyword, 변수, 문자열 등에 따라 색상으로 쉽게 구별 가능
- 강력하고 효율적인 편집기능 : 지능적인 코드 완성, 즉석 오류 검사, 화면 분리
- Debugger : break point 설정, 한 문장씩 수행, 실행 중 변수 값 조회
- 다양한 기능 제공 : 모듈/패키지 설치, 다양한 작업모드(터미널, 파이썬 콘솔) 지원

▣ PyCharm 다운로드 – www.jetbrains.com/pycharm/download

- JetBrains社에서 제작
- python에 특화된 가장 완성도 높은 IDE
- 유료/무료 버전

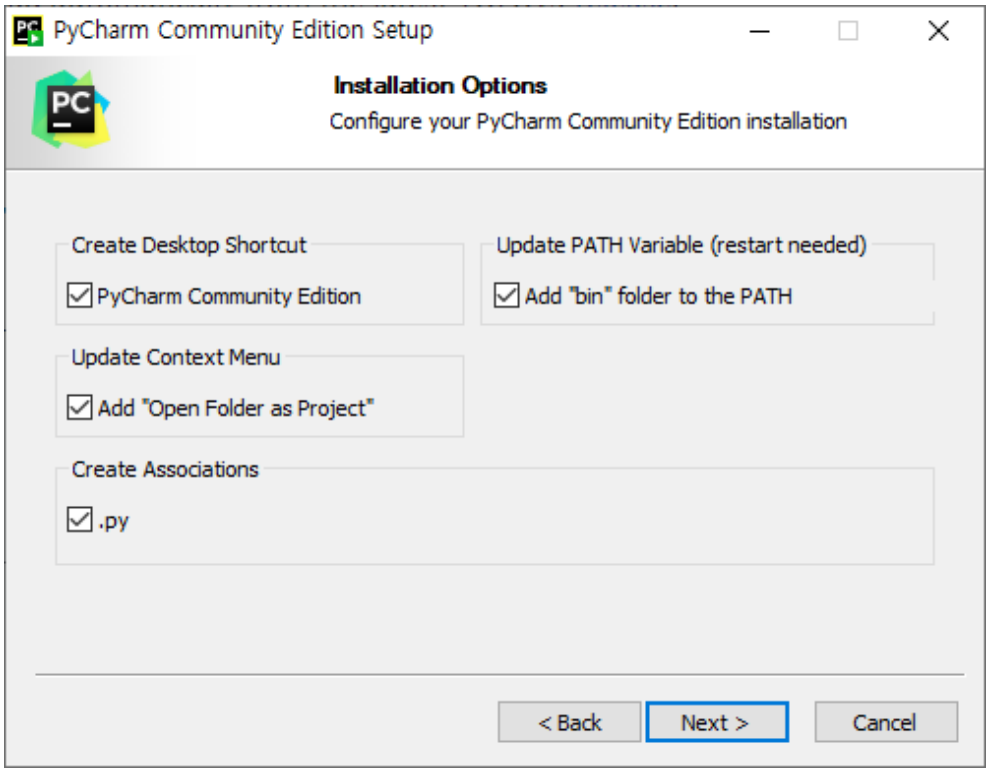


PyCharm 설치 (#2/3)

INDUSTRIAL AI RESEARCH CENTER

▣ PyCharm 설치 (2022.3.1)

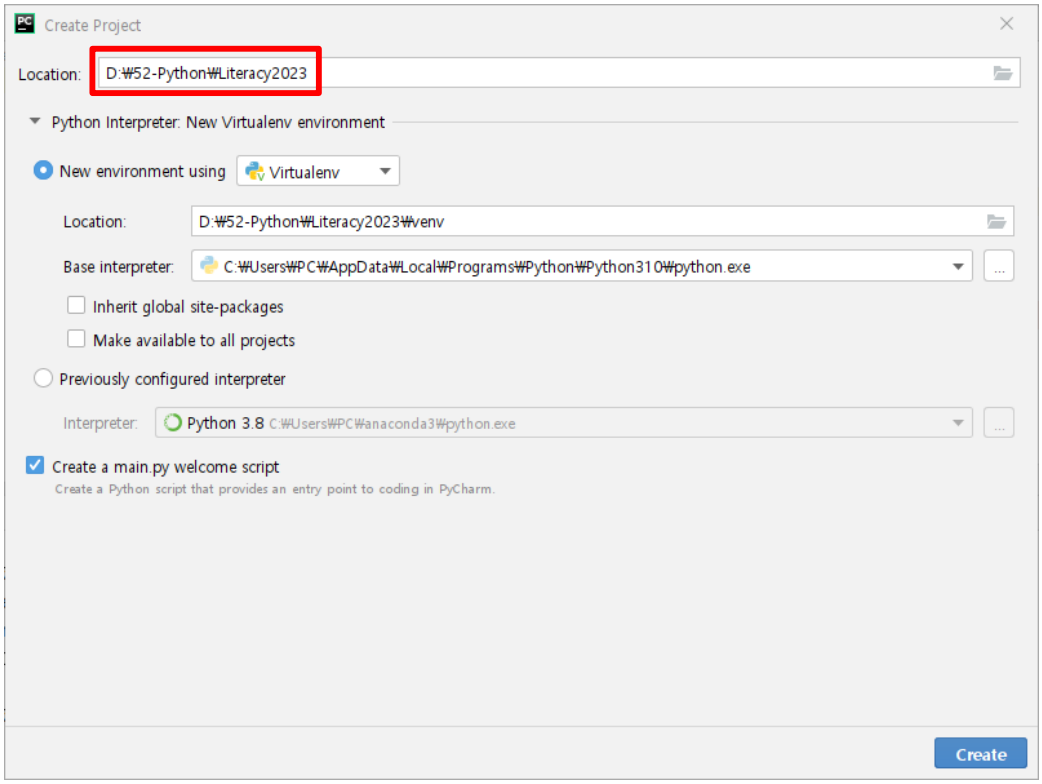
- 설치 옵션 모두 체크



PyCharm 사용법 (#3/3)

INDUSTRIAL AI RESEARCH CENTER

▣ 설치 후 > 새 프로젝트 만들기 : 자동적으로 가상환경을 구축함



YOLOv5 설치 (#1/3)

INDUSTRIAL AI RESEARCH CENTER

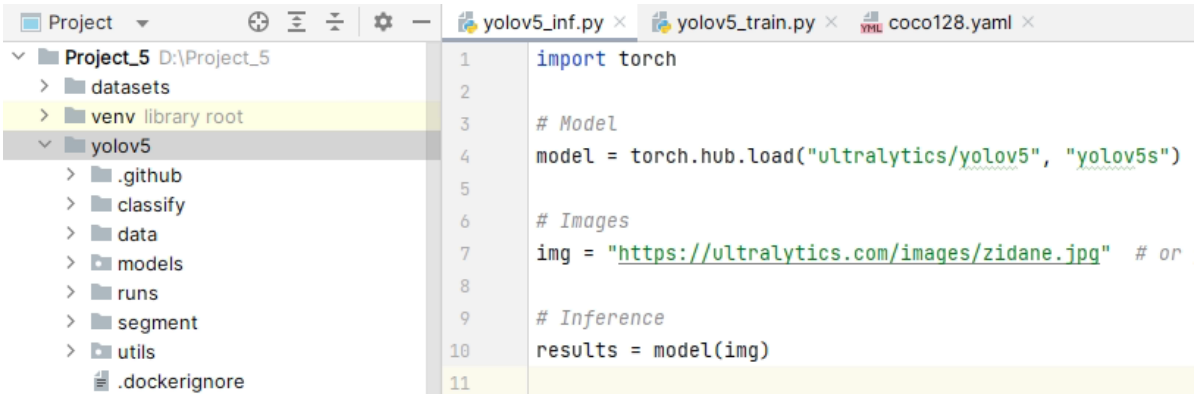
■ 프로젝트 가상환경 만들기, Git 다운로드 및 설치 (https://jhnyang.tistory.com/457)

■ YOLO 설치 → https://github.com/ultralytics/yolov5

▼ Install

Clone repo and install requirements.txt in a Python>=3.7.0 environment, including PyTorch>=1.7.

```
git clone https://github.com/ultralytics/yolov5  # clone
cd yolov5
pip install -r requirements.txt  # install
```



YOLOv5 설치 (#2/3)

INDUSTRIAL AI RESEARCH CENTER

■ YOLO 추론 → https://github.com/ultralytics/yolov5

▼ Inference

YOLOv5 PyTorch Hub inference. Models download automatically from the latest YOLOv5 release.

```
import torch

# Model
model = torch.hub.load("ultralytics/yolov5", "yolov5s")  # or yolov5n - yolov5x6, custom

# Images
img = "https://ultralytics.com/images/zidane.jpg"  # or file, Path, PIL, OpenCV, numpy, list

# Inference
results = model(img)

# Results
results.print()  # or .show(), .save(), .crop(), .pandas(), etc.
```

YOLOv5 설치 (#3/3)

INDUSTRIAL AI RESEARCH CENTER

■ YOLO 학습 → https://github.com/ultralytics/yolov5

▼ Training

The commands below reproduce YOLOv5 COCO results. Models and datasets download automatically from the latest YOLOv5 release. Training times for YOLOv5n/s/m/l/x are 1/2/4/6/8 days on a V100 GPU (Multi-GPU times faster). Use the largest `--batch-size` possible, or pass `--batch-size -1` for YOLOv5 AutoBatch. Batch sizes shown for V100-16GB.

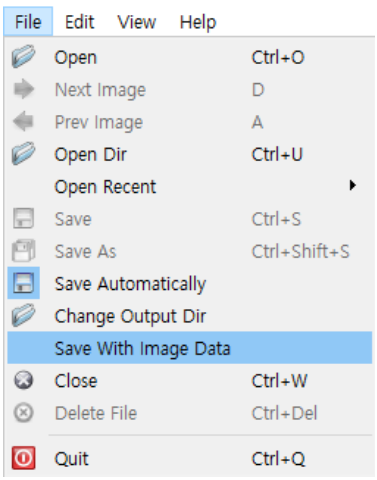
```
python train.py --data coco.yaml --epochs 300 --weights '' --cfg yolov5n.yaml  --batch-size 128
                                                                 yolov5s                64
                                                                 yolov5m                40
                                                                 yolov5l                24
                                                                 yolov5x                16
```

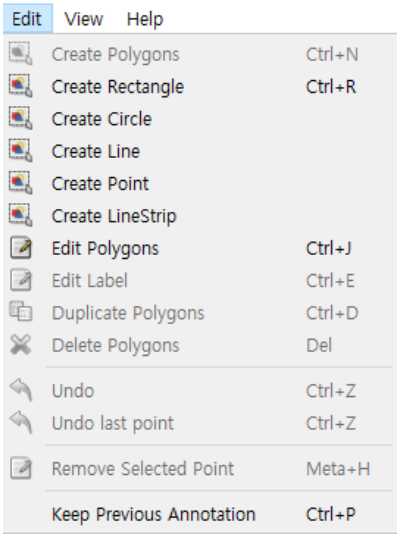
```
(venv) D:\Project_5\yolov5>python train.py --data coco128.yaml --epochs 10 --weights yolov5n.pt  --batch-size 4
train: weights=yolov5n.pt, cfg=, data=coco128.yaml, hyp=data\hyps\hyp.scratch-low.yaml, epochs=10, batch_size=4, imgsz=640, rect=False, resume=
False, nosave=False, noval=False, noautoanchor=False, noplots=False, evolve=None, bucket=, cache=None, image_weights=False, device=, multi_scal
e=False, single_cls=False, optimizer=SGD, sync_bn=False, workers=8, project=runs\train, name=exp, exist_ok=False, quad=False, cos_lr=False, lab
el_smoothing=0.0, patience=100, freeze=[0], save_period=-1, seed=0, local_rank=-1, entity=None, upload_dataset=False, bbox_interval=-1, artifac
t_alias=latest
github: skipping check (offline), for updates see https://github.com/ultralytics/yolov5
requirements: D:\Project_5\requirements.txt not found, check failed.
YOLOv5  v7.0-155-g8ecc727 Python-3.8.10 torch-1.13.1+cu117 CUDA:0 (NVIDIA GeForce GTX 1650 Ti with Max-Q Design, 4096MiB)
```

레이블링 (#1/2)

■ Labelme

- 설치 : pip install labelme
- 특징 : 객체검출(rectangle)과 객체분할(polygon)에 관한 레이블링이 모두 가능
- 사용법
  - 객체검출 : Edit > Create Rectangle
  - 저장옵션 : Save With Image Data 해제





레이블링 (#2/2)

데이터 준비 (#1/2) - prepareDataset

INDUSTRIAL AI RESEARCH CENTER

■ 주석파일의 변환

- Labelme(*.json)로 레이블링한 파일은 Yolo 형식(*.txt)으로 변환

20230326_070007.json - Windows 메모장

파일(F) 편집(E) 서식(O) 보기(V) 도움말(H)


```
{
  "version": "5.0.1",
  "flags": {},
  "shapes": [
    {
      "label": "stamped",
      "points": [
        [
          2120.1438848920866,
          1011.031175059952
        ],
        [
          2113.189448441247,
          1012.2302158273382
        ],
        [
          2109.832134292566,
          1007.4340527577938
        ],
        [
          2111.7505995203837,
          1001.4388489208633
```


Ln 28, Col 10 | 100% | Windows (CRLF) | UTF-8

labels.txt

```
__ignore__
_background_
scratch
stamped
stain
```

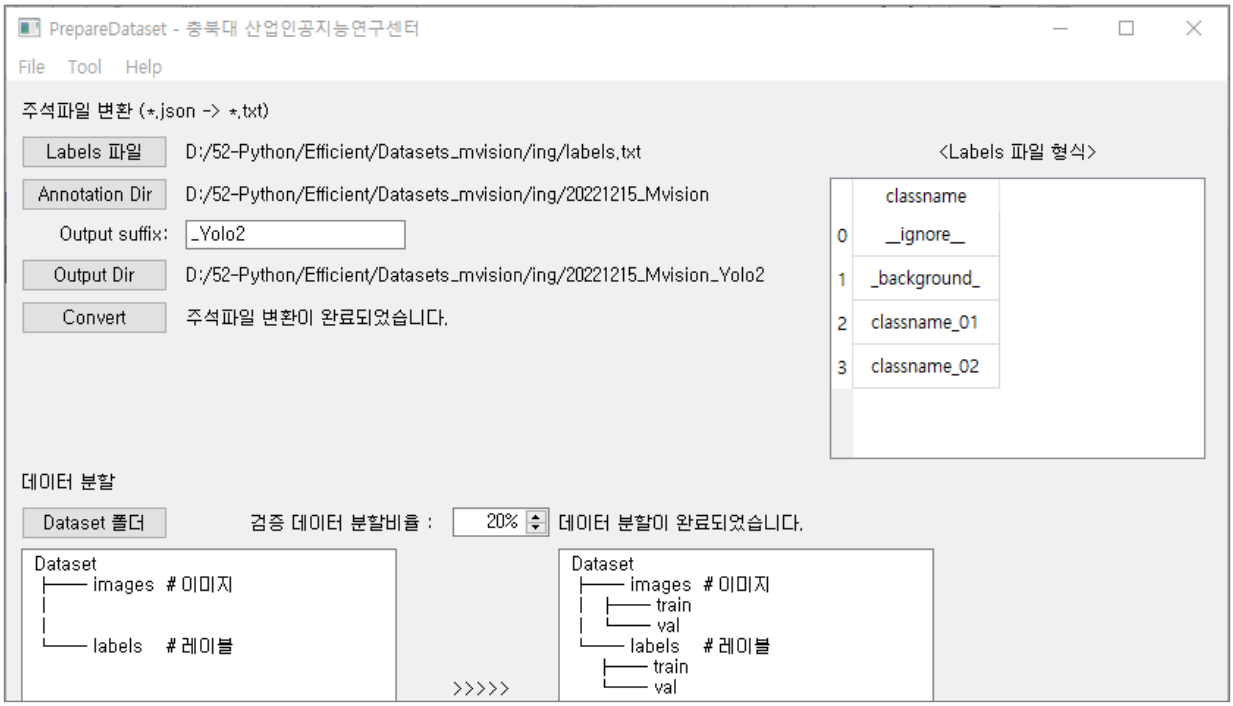
```
0 0.380492 0.409757 0.281559 0.254744
0 0.394802 0.570132 0.217358 0.045380
0 0.213413 0.366440 0.043317 0.040223
```


데이터 준비 (#2/2) - prepareDataset

INDUSTRIAL AI RESEARCH CENTER

■ 데이터 분할

- 전체 데이터를 학습(train)과 검증(val) 데이터로 분할



YOLOv5 주석 형식

INDUSTRIAL AI RESEARCH CENTER

■ YOLOv5의 주석 파일 형식 (*.txt)

- 5개의 값이 공백으로 분리
- (1열) Class ID : **0 (head) / 1(helmet)**
- (2~5열) Bounding box 좌표 : ncx, ncy, nw, nh [0~1]

.../**images**/.../**zidane**.jpg

```
0  0.481719 0.634028 0.690625 0.713278
0  0.741094 0.524306 0.314750 0.933389
27 0.364844 0.795833 0.078125 0.400000
```

.../**lables**/.../**zidane**.txt

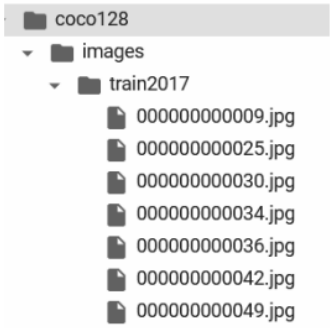
YOLOv5 데이터셋 폴더 구조

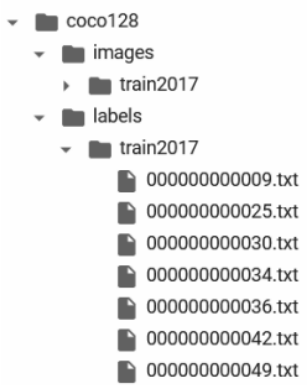
INDUSTRIAL AI RESEARCH CENTER

■ YOLOv5의 Dataset 구조

- 영상별로 주석파일(*.txt)이 매칭

[영상 폴더]





■ 학습용 환경설정파일 (coco128.yaml)

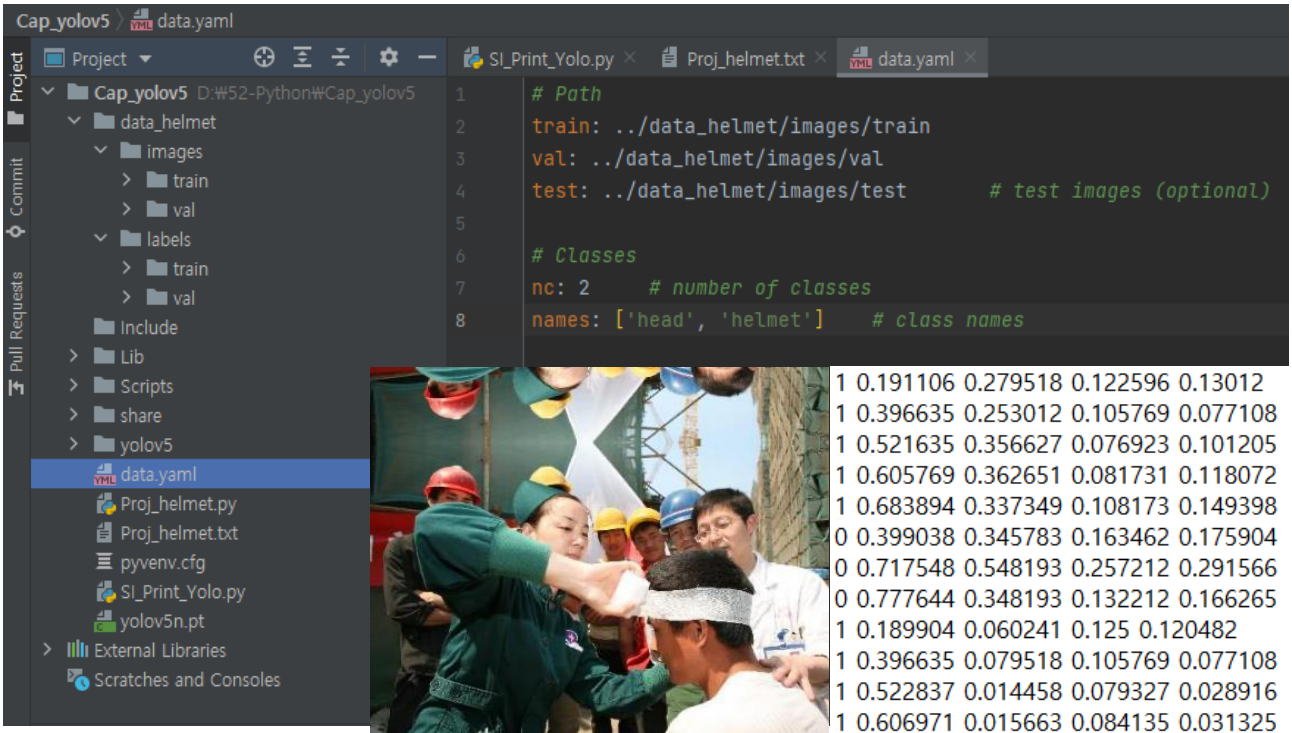
- 영상 폴더는 반드시 'images' 문자열을 포함하도록 지정
- 주석 폴더는 자동적으로 영상폴더에서 'images'만 'labels'로 대체하여 지정됨

```
# download command/URL (optional)
download: https://github.com/ultralytics/yolov5/releas

# train and val data as 1) directory: path/images/, 2)
train: ../coco128/images/train2017/  # 128 images
val: ../coco128/images/train2017/  # 128 images

# number of classes
nc: 80
```

data.yaml 파일 생성

INDUSTRIAL AI RESEARCH CENTER

■ 데이터셋 폴더 구조 생성 → data.yaml 작성 (names는 리스트로 입력하되 ' '는 생략 가능)




```
1 0.191106 0.279518 0.122596 0.13012
1 0.396635 0.253012 0.105769 0.077108
1 0.521635 0.356627 0.076923 0.101205
1 0.605769 0.362651 0.081731 0.118072
1 0.683894 0.337349 0.108173 0.149398
0 0.399038 0.345783 0.163462 0.175904
0 0.717548 0.548193 0.257212 0.291566
0 0.777644 0.348193 0.132212 0.166265
1 0.189904 0.060241 0.125 0.120482
1 0.396635 0.079518 0.105769 0.077108
1 0.522837 0.014458 0.079327 0.028916
1 0.606971 0.015663 0.084135 0.031325
1 0.682692 0.036145 0.110577 0.072289
0 0.40024 0.038554 0.165865 0.077108
0 0.776442 0.03494 0.134615 0.06988
```


학습(1) – 노트북

INDUSTRIAL AI RESEARCH CENTER

■ 학습 – 노트북

- **python train.py** --**data** data.yaml --**weights** yolov5n.pt --**imgsz** 416 --epochs 50 --**batch-size** 16

  - batch-size -1

```
(Cap_yolov5) D:\52-Python\Cap_yolov5>python yolov5\train.py --data data.yaml --weights yolov5n.pt --imgsz 416 --epochs 5 --batch-size 16
train: weights=yolov5n.pt, cfg=, data=data.yaml, hyp=yolov5\data\hyps\hyp.scratch-low.yaml, epochs=5, batch_size=16, imgsz=416, rect=False, resume=False, n
osave=False, noval=False, noautoanchor=False, noplots=False, evolve=None, bucket=, cache=None, image_weights=False, device=, multi_scale=False, single_cls=
False, optimizer=SGD, sync_bn=False, workers=8, project=yolov5\runs\train, name=exp, exist_ok=False, quad=False, cos_lr=False, label_smoothing=0.0, patienc
e=100, freeze=[0], save_period=-1, local_rank=-1, entity=None, upload_dataset=False, bbox_interval=-1, artifact_alias=latest
github: skipping check (offline), for updates see https://github.com/ultralytics/yolov5
YOLOv5  v6.1-246-g2dd3db0 Python-3.8.10 torch-1.9.0+cu102 CUDA:0 (NVIDIA GeForce GTX 1650 Ti, 4096MiB)
```

...

```
     Epoch   gpu_mem       box       obj       cls    labels  img_size
       4/4     0.94G   0.04403   0.02723   0.00493       120       416: 100%|██████████| 250/250 [11:22<00:00,  2.73s/it]
                Class     Images     Labels          P          R     mAP@.5 mAP@.5:.95: 100%|██████████| 32/32 [00:30<00:00,  1.05it/s]
                  all       1000       4966      0.869      0.786      0.857      0.464

5 epochs completed in 0.984 hours.
```

■ 테스트 결과

```
Validating yolov5\runs\train\exp3\weights\best.pt...
Fusing layers...
Model summary: 213 layers, 1761871 parameters, 0 gradients, 4.2 GFLOPs
                Class     Images     Labels          P          R     mAP@.5 mAP@.5:.95: 100%|██████████| 32/32 [00:33<00:00,  1.04s/it]
                  all       1000       4966      0.869      0.786      0.857      0.464
                 head       1000       1185      0.858       0.75      0.824      0.415
               helmet       1000       3781       0.88      0.822       0.89      0.513
Results saved to yolov5\runs\train\exp3
```

학습(2) – 학습용 서버

INDUSTRIAL AI RESEARCH CENTER

■ 학습 – 개인용 서버

- **python train.py** --**data** data.yaml --**weights** yolov5n.pt --**imgsz** 416 --**epochs** 50 --**batch-size** 32

```
autobatch: Computing optimal batch size for --imgsz 416
autobatch: CUDA:0 (GeForce RTX 3090) 24.00G total, 0.04G reserved, 0.02G allocated, 23.94G free
      Params      GFLOPs  GPU_mem (GB)  forward (ms) backward (ms)                   input
     1766623       1.766         0.075         26.68         20.96        (1, 3, 416, 416)
     1766623       3.532         0.113         15.93         19.62        (2, 3, 416, 416)
     1766623       7.065         0.193         17.68         20.46        (4, 3, 416, 416)
     1766623       14.13         0.348         18.46         14.63        (8, 3, 416, 416)
     1766623       28.26         0.700         23.71         14.14       (16, 3, 416, 416)
autobatch: Using batch-size 516 for CUDA:0 21.60G/24.00G (90%)
```

■ 시스템 사양

```
YOLOv5  v6.0-76-g79bca2b torch 1.9.0+cu111 CUDA:0 (GeForce RTX 3090, 24576MiB)
```

```
     Epoch   gpu_mem       box       obj       cls    labels  img_size
     49/49     1.1G   0.03109   0.02232  0.001484       217       416: 100%|██████████| 125/125 [00:12<00:00, 10.00it/s]
               Class     Images     Labels          P          R     mAP@.5 mAP@.5:.95: 100%|██████████| 16/16 [00:03<00:00,  4.73it/s]
                 all       1000       4966      0.924      0.862      0.922      0.583

50 epochs completed in 0.228 hours.
```


[참고] 모델 개발

INDUSTRIAL AI RESEARCH CENTER

■ 학습

- cd yolov5
- python train.py --**data data.yaml** --**weights yolov5l6.pt** --**imgsz 1280** --**epochs 300** --**batch-size 8**

```
python yolov5/train.py --data D:\02-Mvision\DataMvision\4-Mvision_Yolo\data.yaml --weights yolov5l6.pt --imgsz 1280 --batch-size 8 --epochs 300 --
project DataMvision/4-Mvision_Yolo_results --name exp01_l6 --patience 0

train: weights=yolov5l6.pt, cfg=, data=D:\02-Mvision\DataMvision\4-Mvision_Yolo\data.yaml, hyp=yolov5\data\hyps\hyp.scratch-low.yaml, epochs=300,
batch_size=8, imgsz=1280, rect=False, resume=False, nosave=False, noval=False, noautoanchor=False, noplots=False, evolve=None, bucket=, cache=None
, image_weights=False, device=, multi_scale=False, single_cls=False, optimizer=SGD, sync_bn=False, workers=8, project=DataMvision/4-
Mvision_Yolo_results, name=exp01_l6, exist_ok=False, quad=False, cos_lr=False, label_smoothing=0.0, patience=0, freeze=[0], save_period=-1, seed=0
, local_rank=-1, entity=None, upload_dataset=False, bbox_interval=-1, artifact_alias=latest
github: skipping check (offline), for updates see https://github.com/ultralytics/yolov5
YOLOv5  v7.0-46-g96a71b1 Python-3.9.11 torch-1.13.1+cu116 CUDA:0 (NVIDIA GeForce RTX 3090, 24576MiB)
```

```
300 epochs completed in 1.739 hours.
Optimizer stripped from DataMvision\4-Mvision_Yolo_results\exp01_l6\weights\last.pt, 153.6MB
Optimizer stripped from DataMvision\4-Mvision_Yolo_results\exp01_l6\weights\best.pt, 153.6MB

Validating DataMvision\4-Mvision_Yolo_results\exp01_l6\weights\best.pt...
Fusing layers...
Model summary: 346 layers, 76134048 parameters, 0 gradients, 109.9 GFLOPs
                 Class     Images  Instances          P          R      mAP50   mAP50-95: 100%|██████████|
                   all        103       1970      0.861      0.832       0.86      0.475
               scratch        103        251      0.908      0.821      0.876       0.63
               stamped        103       1337      0.717      0.716      0.738      0.266
                 stain        103        382      0.959      0.958      0.965       0.53
Results saved to DataMvision\4-Mvision_Yolo_results\exp01_l6
```

학습 Tip

INDUSTRIAL AI RESEARCH CENTER

■ Tips for Best Training Results

Most of the time good results can be obtained with no changes to the models or training settings, **provided your dataset is sufficiently large and well labelled**. If at first you don't get good results, there are steps you might be able to take to improve, but we always recommend users **first train with all default settings** before considering any changes. This helps establish a performance baseline and spot areas for improvement.

Dataset

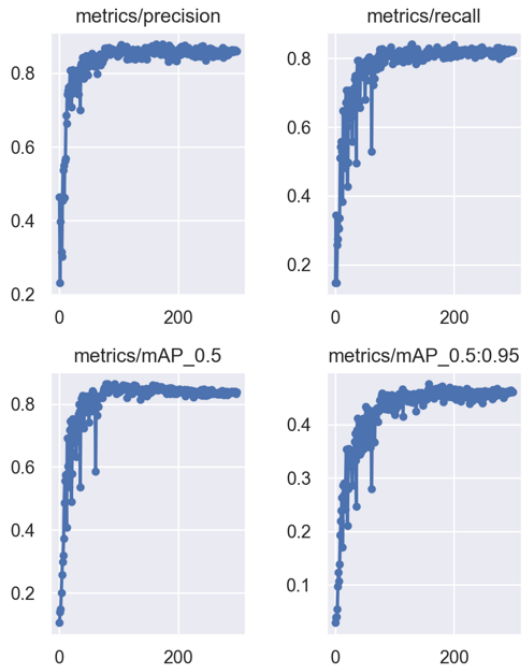
- **Images per class.** ≥ 1500 images per class recommended
- **Instances per class.** ≥ 10000 instances (labeled objects) per class recommended
- **Image variety.** Must be representative of deployed environment. For real-world use cases we recommend images from different times of day, different seasons, different weather, different lighting, different angles, different sources (scraped online, collected locally, different cameras) etc.
- **Label consistency.** All instances of all classes in all images must be labelled. Partial labelling will not work.
- **Label accuracy.** Labels must closely enclose each object. No space should exist between an object and it's bounding box. No objects should be missing a label.
- **Label verification.** View `train_batch*.jpg` on train start to verify your labels appear correct, i.e. see example mosaic.
- **Background images.** Background images are images with no objects that are added to a dataset to reduce False Positives (FP). We recommend about 0-10% background images to help reduce FPs (COCO has 1000 background images for reference, 1% of the total). No labels are required for background images.

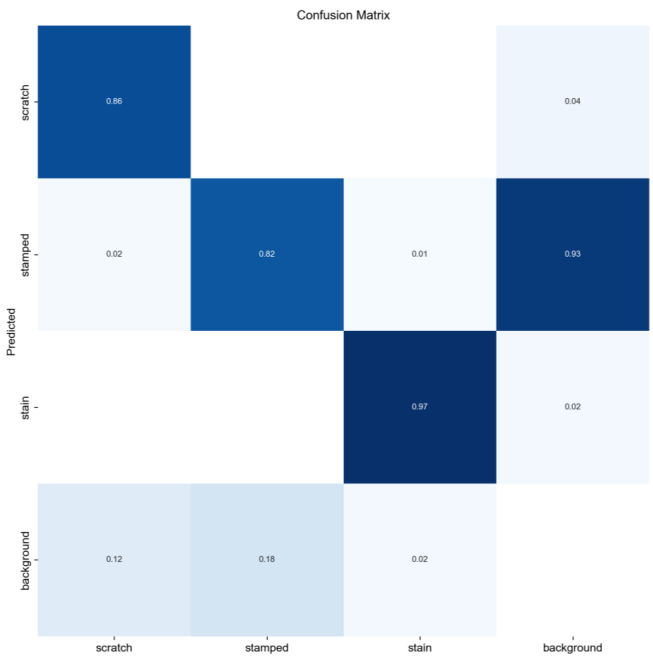
학습 결과 (#1/2)

INDUSTRIAL AI RESEARCH CENTER

■ 학습 과정 및 결과

- epoch에 따른 평가지표의 추이
- confusion matrix 등





학습 결과 (#2/2)

INDUSTRIAL AI RESEARCH CENTER

■ 검출 사례

추론(inference)

INDUSTRIAL AI RESEARCH CENTER

■ **python detect.py --source** path₩to₩image **--weights** weights₩best.pt
--imgsz 416 **--conf_thres** 0.45(default)

image 1003/1003 C:\yolov5\data_helmet\images\test\helmet_1003.jpg: 352x416 7 helmets, Done. (0.009s)
Speed: 0.3ms pre-process, 8.5ms inference, 1.0ms NMS per image at shape (1, 3, 416, 416)

```
0 0.553889 0.133333 0.0788889 0.136667 0.697266
1 0.163333 0.325833 0.304444 0.418333 0.887207
1 0.736667 0.438333 0.195556 0.303333 0.897461
```

■ **python val.py --data** data.yaml --**weights** runs₩train₩exp28₩weights₩best.pt
--**imgsz** 416 --**task** test --**conf_thres** 0.001(default)

```
               Class     Images     Labels          P          R     mAP@.5 mAP@.5:.95: 100%
                 all       1003       4331       0.92       0.78      0.852      0.459
                head       1003        997      0.956      0.638      0.764      0.392
              helmet       1003       3334      0.884      0.923      0.939      0.527
Speed: 0.0ms pre-process, 0.5ms inference, 1.0ms NMS per image at shape (32, 3, 416, 416)
```

```
1 0.163959 0.321751 0.290012 0.435018 0.913086
1 0.734657 0.436259 0.193742 0.300767 0.90625
0 0.554302 0.132559 0.0803249 0.140569 0.625488
1 0.55355 0.133348 0.0794224 0.142148 0.0404358
0 0.16505 0.322428 0.287229 0.439079 0.00555038
1 0.201564 0.0533619 0.222022 0.106724 0.00437927
1 0.398616 0.335966 0.0451263 0.0726535 0.00389671
0 0.555505 0.139892 0.100782 0.190884 0.00289345
0 0.738869 0.437274 0.222623 0.307762 0.00240517
0 0.963297 0.479242 0.0493382 0.198556 0.00188541
0 0.536703 0.164034 0.066787 0.138087 0.00185108
1 0.609507 0.273014 0.0637785 0.185469 0.00156212
1 0.693141 0.430505 0.122744 0.268953 0.00150776
0 0.979844 0.836868 0.0403129 0.245036 0.0012455
1 0.089839 0.315208 0.13741 0.343412 0.0011816
1 0.216832 0.0311372 0.193291 0.0622744 0.00116539
0 0.590403 0.162906 0.0682912 0.138989 0.00112247
```

(모든 confidence의 결과를 포함할 것)

감사합니다

Q & A