

[프로젝트 #2]

# YOLO를 이용한 안전모 검출

2022. 06. 08

김 현 용

충북대학교 산업인공지능학과

# 목 차

1. YOLOv5 환경구성
2. 데이터 준비
3. 학습
4. 학습 결과
5. 검증
6. 테스트

# YOLOv5 환경구성(1) - 실패

- 가상환경 만들기

- Python 3.8

- YOLOv5 설치

- Warning 무시

▼ 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
```

- Pytorch 설치

- pip uninstall torch # PermissionError: [WinError 32] → IDE를 관리자권한으로 재실행

- import torch

torch.\_\_version\_\_

Out: '1.10.0+cu102'

- 그래도 결국 안됨  
(깔끔하게 다시)

PyTorch Build	Stable (1.11.0)		Preview (Nightly)		LTS (1.8.2)	
Your OS	Linux		Mac		Windows	
Package	Conda	Pip		LibTorch	Source	
Language	Python			C++ / Java		
Compute Platform	CUDA 10.2	CUDA 11.3		ROCm 4.5.2 (beta)	CPU	
Run this Command:	<pre>pip3 install torch torchvision torchaudio --extra-index-url https://download.pytorch.org/whl/cu113</pre>					

# YOLOv5 환경구성(2) - 실패

- 가상환경 만들기

  - Python 3.8

- Pytorch 설치

  - import torch  
torch.\_\_version\_\_  
Out: '1.10.0+cu102'

PyTorch Build	Stable (1.11.0)	Preview (Nightly)	LTS (1.8.2)	
Your OS	Linux	Mac	Windows	
Package	Conda	Pip	LibTorch	Source
Language	Python		C++ / Java	
Compute Platform	CUDA 10.2	CUDA 11.3	ROCm 4.5.2 (beta)	CPU
Run this Command:	pip3 install torch torchvision torchaudio --extra-index-url https://download.pytorch.org/whl/cu113			

- YOLOv5 설치

  - Warning 무시

## ▼ 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 환경구성(3) - 실패

- 추론은 잘 되지만, 학습 시 다음과 같은 문제가 발생
  - box, obj, cls가 nan으로 나타남

```
Epoch    gpu_mem    box    obj    cls    labels  img_size
  0/4      0.95G     nan    nan    nan      71     640: 100%|██████████| 134/134 [03:31<00:00, 1.58s/it]
```

cuda 11.3, GTX 1660, Windows 10 환경에서도 문제를 만났습니다.

cuda 11.3을 10.2로 바꿔서 해결했습니다.

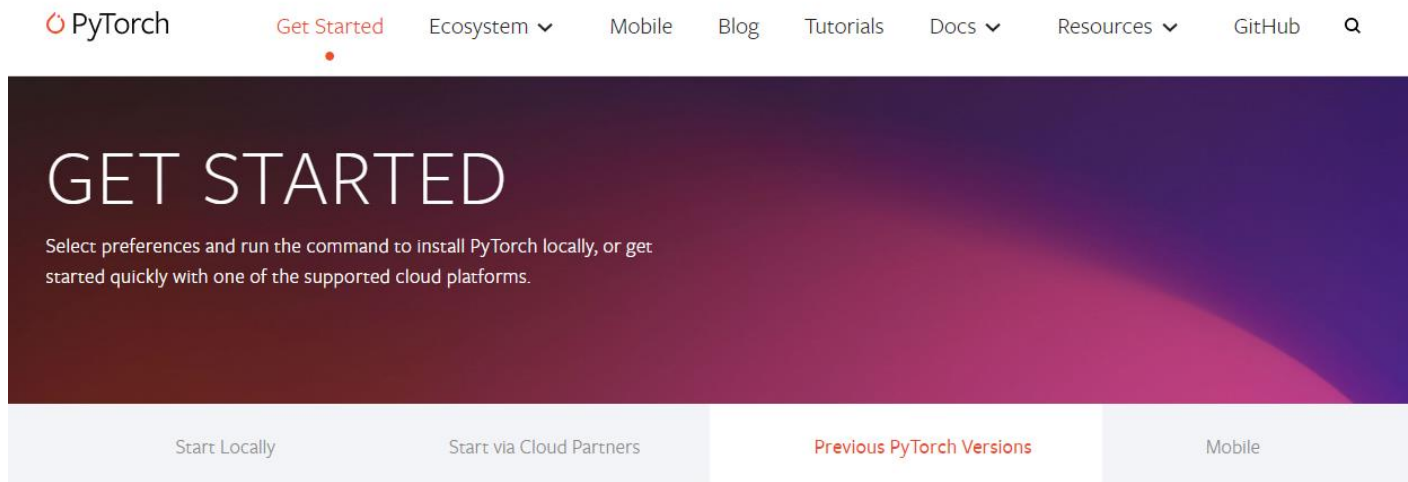
conda remove --n "my env name" --all로 완전히 제거

하고 cuda 11.3 관련 프로그램을 제거했습니다.

conda virtual env를 다시 만들고 cuda 10.2 및 cuDNN을 포함한 모든 프로그램을 다시 설치하십시오.  
도움이 되기를 바랍니다.

- Nvidia driver / CUDA toolkit / cuDNN 버전 호환성 문제

<https://velog.io/@boom109/nvidia-driver-cuda-toolkit-cudnn-install>

The image shows the top section of the PyTorch website. At the top is a navigation bar with the PyTorch logo, a 'Get Started' link (highlighted with a red dot), and other links like 'Ecosystem', 'Mobile', 'Blog', 'Tutorials', 'Docs', 'Resources', and 'GitHub'. Below the navigation bar is a large banner with a purple-to-pink gradient. The banner features the text 'GET STARTED' in large white letters, followed by a smaller line of text: 'Select preferences and run the command to install PyTorch locally, or get started quickly with one of the supported cloud platforms.' At the bottom of the banner are four buttons: 'Start Locally', 'Start via Cloud Partners', 'Previous PyTorch Versions', and 'Mobile'.

# YOLOv5 환경구성(4) – 성공(1)

- Pytorch 설치

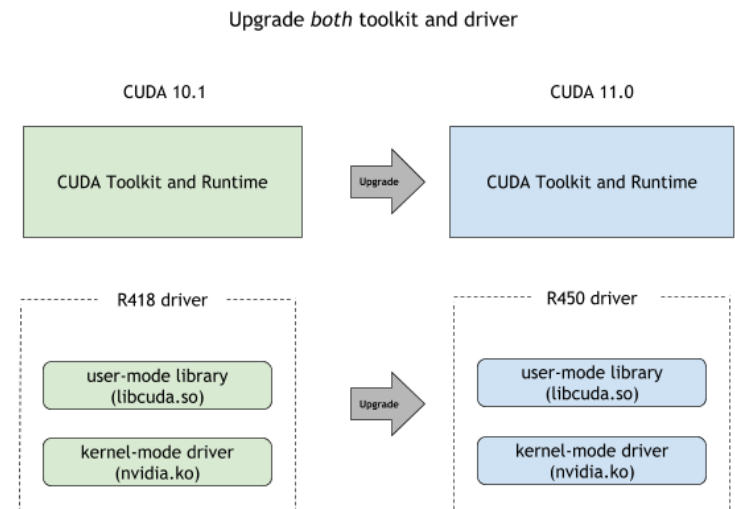
- 노트북: 'cuda 11.9.0+cu102'

pip3 install torch==1.9.0+cu102 torchvision==0.10.0+cu102 torchaudio===0.9.0 -f

[https://download.pytorch.org/whl/torch\\_stable.html](https://download.pytorch.org/whl/torch_stable.html)

- 학습용 서버 : cu11.1

PyTorch Build	Stable (1.9.0)		Preview (Nightly)	LTS (1.8.2)
Your OS	Linux		Mac	Windows
Package	Conda	Pip	LibTorch	Source
Language	Python		C++ / Java	
Compute Platform	CUDA 10.2	CUDA 11.1	ROCm 4.2 (beta)	CPU
Run this Command:	pip3 install torch==1.9.0+cu102 torchvision==0.10.0+cu102 torchaudio===0.9.0 -f https://download.pytorch.org/whl/torch_stable.html			



# YOLOv5 환경구성(5) – 성공(2)

- Pytorch 설치

- import torch  
torch.\_\_version\_\_  
Out: '1.8.2+cu102'

Long Term Support (장기지원)

Stable (1.11.0)		Preview (Nightly)		LTS (1.8.2)
Linux		Mac		Windows
Conda	Pip	LibTorch	Source	
Python		C++ / Java		
CUDA 10.2	CUDA 11.1	ROCm 4.5.2 (beta)	CPU	

```
pip3 install torch==1.8.2 torchvision==0.9.2 torchaudio==0.8.2 --extra-index-url https://download.pytorch.org/whl/lts/1.8/cu102
```

**Note:** Additional support for these binaries may be provided by PyTorch Enterprise Support Program Participants.

- YOLOv5 설치

- Clone repo and install [requirements.txt](#) in a **Python>=3.7.0** environment, including **PyTorch>=1.7**.

- 명령어

```
git clone https://github.com/ultralytics/yolov5
```

```
cd yolov5
```

```
pip install -r requirements.txt
```

# 데이터 준비(1) - 주석 파일 변환

- 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





# 데이터 준비(2) – 폴더 구조 생성


- YOLOv5 폴더 구조 생성

Cap\_yolov5 > data.yaml

Project  
▼ Cap\_yolov5 D:\#52-Python#Cap\_yolov5  
▼ data\_helmet  
▼ images  
    > train  
    > val  
▼ labels  
    > train  
    > val  
Include  
> Lib  
> Scripts  
> share  
> yolov5  
data.yaml  
Proj\_helmet.py  
Proj\_helmet.txt  
pyenv.cfg  
SI\_Print\_Yolo.py  
yolov5n.pt  
External Libraries  
Scratches and Consoles

SI\_Print\_Yolo.py Proj\_helmet.txt data.yaml

```
1 # Path
2 train: ../data_helmet/images/train
3 val: ../data_helmet/images/val
4 test: ../data_helmet/images/test # test images (optional)
5
6 # Classes
7 nc: 2 # number of classes
8 names: ['head', 'helmet'] # class names
```



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

- YOLOv5의 Dataset 구조

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

```
(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	0.946	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%
	all	1000	4966	0.869	0.786	0.857	0.464

5 epochs completed in 0.984 hours.

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%
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

- YOLOv5의 Dataset 구조

- **python train.py --data data.yaml --weights yolov5n.pt --imgsz 416 --epochs 50**  
**--batch-size 32**  
**--batch-size -1**

```
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

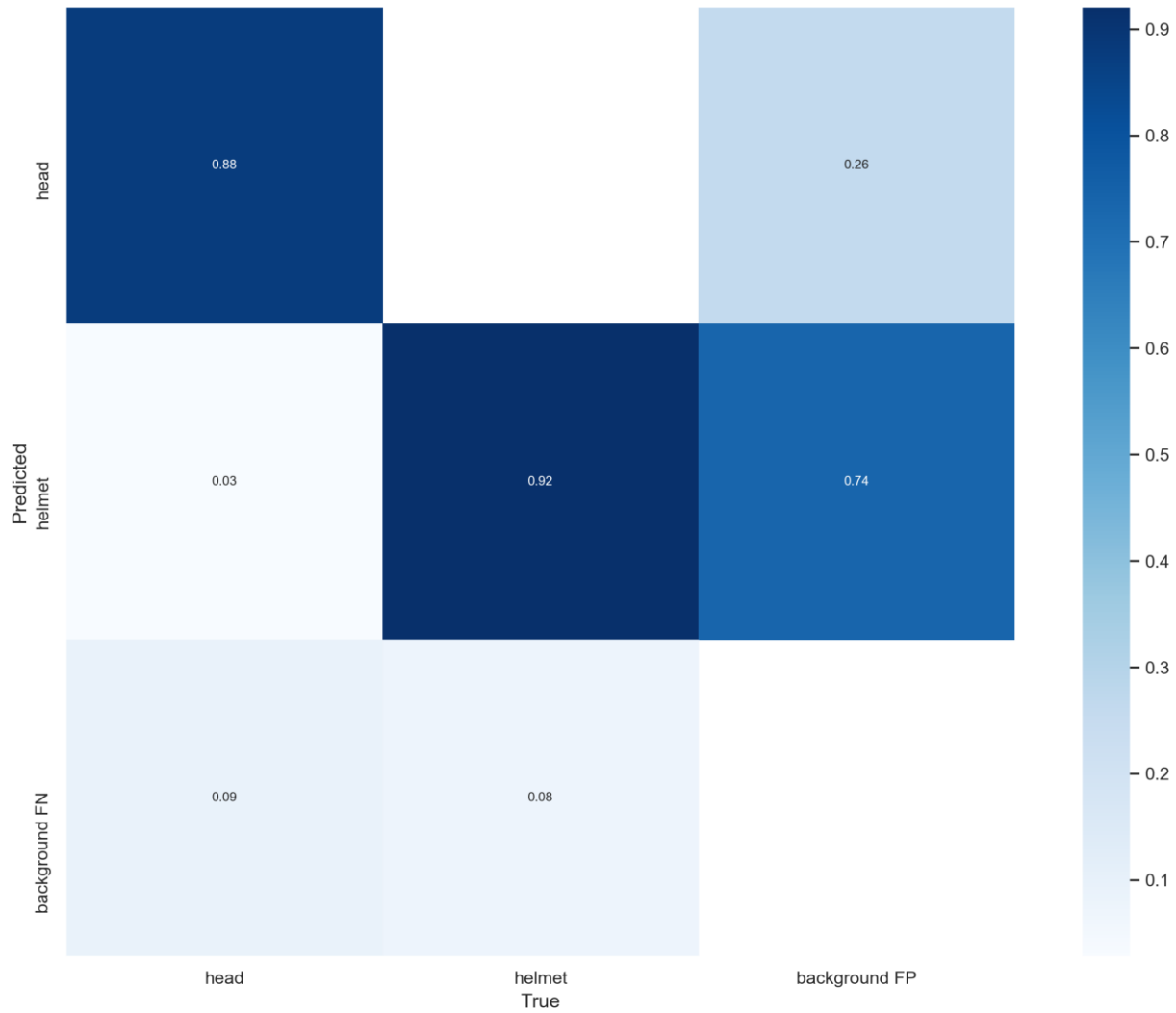
```
YOLOv5 v6.0-76-g79bca2b torch 1.9.0+cu111 CUDA:0 (GeForce RTX 3090, 24576M1B)
```

```
Epoch  gpu_mem    box    obj    cls  labels  img_size
49/49   1.16  0.03189  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.
```

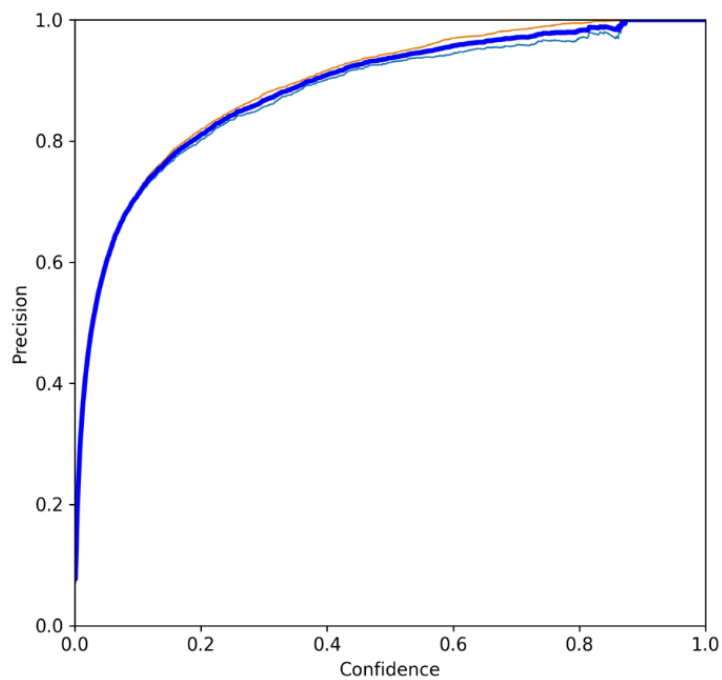
# 학습 결과

- Confusion Matrix

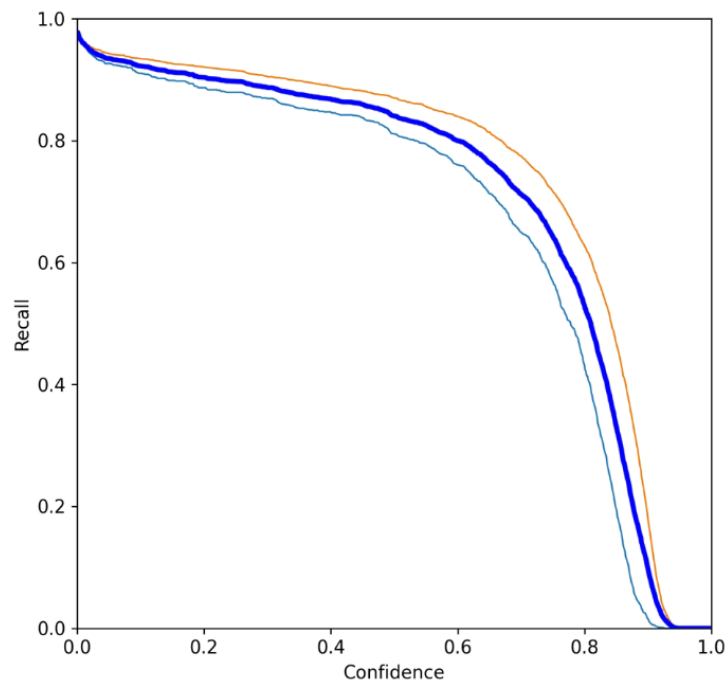


# 학습 결과

- PR Curve



— head  
— helmet  
— all classes 1.00 at 0.875



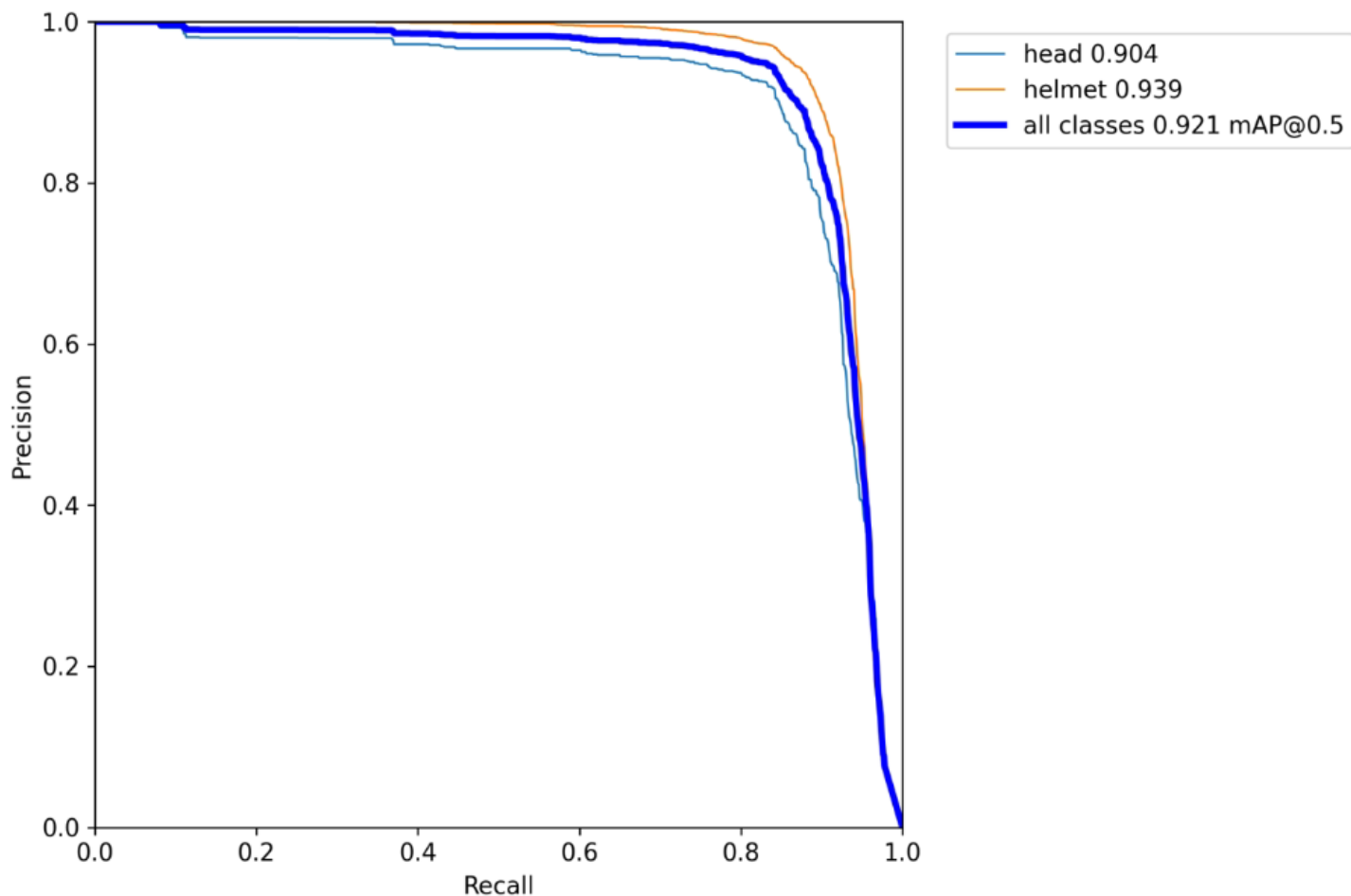
— head  
— helmet  
— all classes 0.98 at 0.000

# 학습 결과

- PR Curve

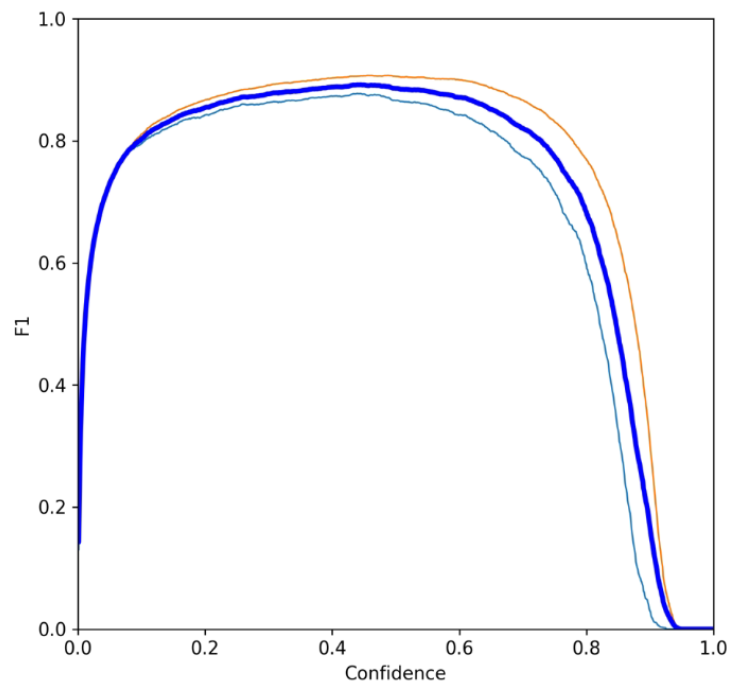
Model Summary: 213 layers, 1761871 parameters, 0 gradients, 4.2 GFLOPs

Class	Images	Labels	P	R	mAP@.5	mAP@.5:.95:
all	1000	4966	0.925	0.861	0.921	0.583
head	1000	1185	0.918	0.84	0.904	0.565
helmet	1000	3781	0.931	0.883	0.939	0.6



# 학습 결과

- PR Curve



— head  
— helmet  
**all classes 0.89 at 0.444**

metrics/precision



metrics/recall



metrics/mAP\_0.5



metrics/mAP\_0.5:0.95





# 학습 결과

- 검출 사례





# 검증(테스트)

- `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
```

# 추론

- `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
```



# 결과 제출방법

- 테스트 데이터셋에 대한 **예측값** 제출방법

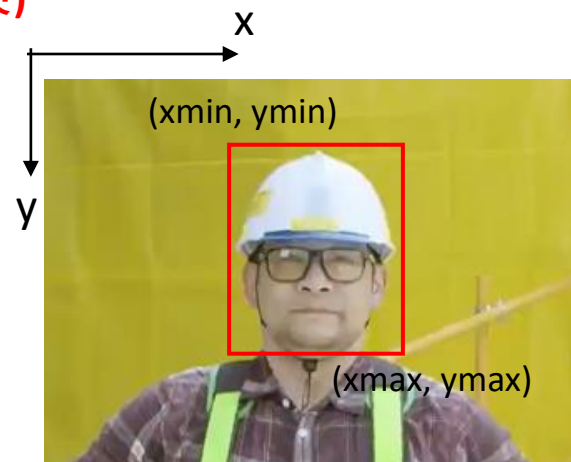
- 각 이미지와 동일한 이름의 텍스트 파일로 저장 (예, abc.png → abc.txt)

- 텍스트(.txt)파일의 예측값은 다음 형식으로 작성

**<class name> <confidence(0~1)> <xmin(px)> <ymin(px)> <xmax(px)> <ymax(px)>**

예측값 파일의 예 **(모든 confidence의 결과를 포함할 것)**

```
head 0.460851 429 219 528 247
helmet 0.287150 336 231 376 305
helmet 0.292345 0 199 88 436
head 0.269833 433 260 506 336
```



- 평가지표 : mAP@IoU=0.5

- Object detection에서 주로 사용하는 평가방법([PASCAL VOC 2012 competition](https://paperswithcode.com/paper/pascal-voc-2012-competition))

- IoU > 0.5인 경우에만 검출되었다고 판단

- 계산방법 참조 : <https://github.com/Cartucho/mAP>

# 평가지표 및 심사방법

---

- 평가지표 :  $mAP@IoU=0.5$ 
  - Object detection에서 주로 사용하는 평가방법([PASCAL VOC 2012 competition](#))
  - $IoU > 0.5$ 인 경우에만 검출되었다고 판단
  - 계산방법 참조 : <https://github.com/Cartucho/mAP>
- 심사방법
  - 심사위원장 1명 + 심사위원 2명
    - 황영배 (충북대학교 지능로봇공학과 교수)
    - 김현용 (충북대학교 산업인공지능연구센터 초빙교수)
    - 김현호 (충북대학교 산업인공지능연구센터 초빙교수)
  - 평가지표를 계산한 후 순위권 팀에 대해 **사후검증(예측값 재현여부)** 실시
  - 입상팀은 시상식(11.2) 때 모델 개발에 대한 **브리핑(5~10분)** 실시 예정

# 수상팀 및 특징

순위	팀명	정확도 (mAP@0.5)	개발 방법	
			데이터	모델
금	Tensor	<b>80.98%</b>	추가 <sup>1)</sup> , 증량 <sup>2)</sup>	YOLOv5l
은	MSISLAB	80.29%	증량	YOLOv4tiny
	투피스	79.67%	재라벨링 <sup>3)</sup> , 추가, 증량	YOLOv5x
동	MSIS	76.13%	재라벨링	YOLOv5l
	LAB423	76.04%	추가, 증량	YOLOv5x + VGG

1) 추가 : 대회에서 제공한 데이터 외의 데이터를 추가한 경우

2) 증량 : 데이터에 Augmentation 기법을 적용하여 데이터 수를 증가시킨 경우

3) 재라벨링 : 대회에서 제공한 데이터의 라벨링을 대회 취지에 맞게 수정한 경우

**Q & A**