

# YOLOv5 사용법

2022. 5. 18

산업인공지능학과

2020254014 임동민, 2020254020 안건호, 2020254019 신정환



**충북대학교**  
CHUNGBUK NATIONAL UNIVERSITY

**INDAI** INDUSTRIAL AI  
RESEARCH CENTER

# Agenda

---

- 1. Colab에서 환경구축**
- 2. 학습 파라미터 설정**
- 3. 학습/검증/추론 방법**
- 4. 실습**
- 5. Q&A**

# Colab에서 환경구축

- Google colab에 접속하고 새 노트를 생성
- 런타임-런타임 유형 변경을 선택 후, 가속기를 GPU로 설정
- yolov5 파일을 다운로드 및 필수 라이브러리를 설치

```
!git clone https://github.com/ultralytics/yolov5          # yolov5 코드 clone
%cd yolov5                                                # clone한 폴더로 진입
%pip install -qr requirements.txt                        # 필수 라이브러리 설치
```

- Custom dataset 업로드
- 데이터 셋 파일 압축 해제

```
!unzip ../custom_dataset.zip
```

# Colab에서 환경구축

- ` yolov5/data/` 폴더에 dataset.yaml` 파일 작성

```
path: /content/yolov5m/dataset
train: train/images
val: valid/images
test: test/images

nc: 2
names: ['mask', 'no-mask']
```

# 학습 파라미터 설정

## - parse\_opt 함수 내 파라미터 정의

```
def parse_opt(known=False):
    parser = argparse.ArgumentParser()
    parser.add_argument('--weights', type=str, default=ROOT / 'yolov5s.pt', help='initial weights path')
    parser.add_argument('--cfg', type=str, default='', help='model.yaml path')
    parser.add_argument('--data', type=str, default=ROOT / 'data/coco128.yaml', help='dataset.yaml path')
    parser.add_argument('--hyp', type=str, default=ROOT / 'data/hyps/hyp.scratch.yaml', help='hyperparameters path')
    parser.add_argument('--epochs', type=int, default=300)
    parser.add_argument('--batch-size', type=int, default=16, help='total batch size for all GPUs, -1 for autobatch')
    parser.add_argument('--imgsz', '--img', '--img-size', type=int, default=640, help='train, val image size (pixels)')
    parser.add_argument('--rect', action='store_true', help='rectangular training')
```

Yolo를 학습하기 위해서 설정하는 파라미터는 약 40여가지가 됨  
그 중 자주 쓰이는 것 위주로 소개

# 학습 파라미터 설정

## 1. 파라미터 설명

### 1) 입력 이미지 크기 설정

- YOLOv5는 학습할 때 모든 이미지 사이즈를 동일하게 정사각형으로 resizing
- Default값은 640 (640은 640x640픽셀을 의미)
- 이미지 사이즈를 크게 설정할수록 모델 성능은 더 좋아질 수 있음
- 대신 학습속도와 리소스 부담이 커지는 단점
- 주의점으로는 검증이나 시험할 때 학습에 사용한 이미지 사이즈와 동일하게 설정
- 이미지 크기 설정하는 방법

```
--imgsz(or --img or --img-size) 값
```

### 2) 배치 사이즈 설정

- 학습할 때 한번에 처리할 이미지 수를 지정할 수 있음. Default값은 16
- 배치 사이즈 설정하는 방법

```
--batch-size 값
```

# 학습 파라미터 설정

## 3) 에포크 수 설정

- 데이터셋으로 학습을 반복할 횟수를 지정할 수 있다. Default값은 300
- 에포크 수 설정

```
--epochs 값
```

## 4) 데이터 경로 설정

- 데이터셋 정보가 적힌 yaml파일의 경로를 설정

```
--data "dataset.yaml" (예시)
```

## 5) 모델 아키텍처 경로 설정

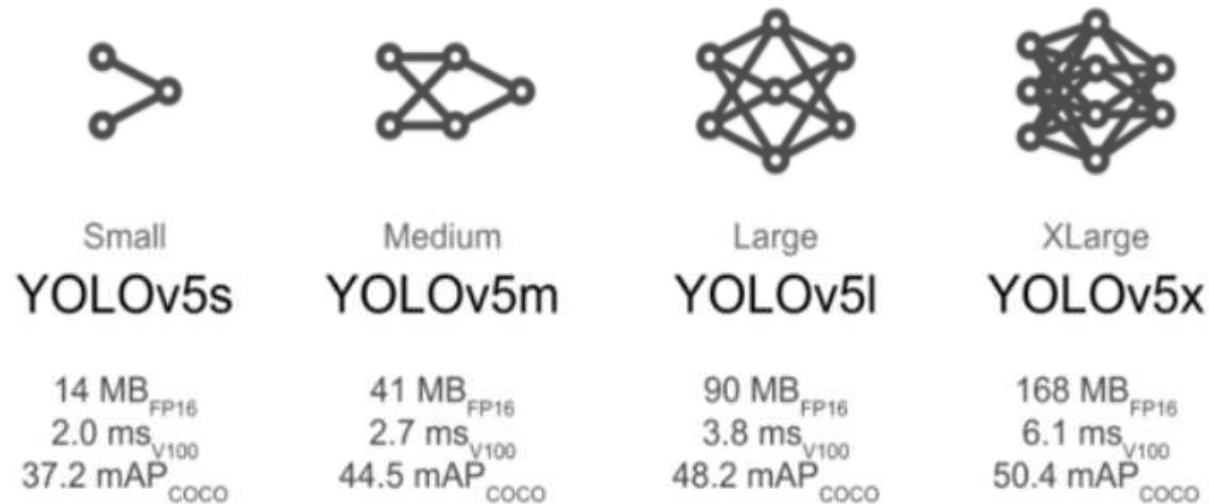
- YOLOv5 아키텍처 yaml 파일의 경로를 설정

```
--cfg 'models/yolov5m.yaml'(예시)
```

# 학습 파라미터 설정

## 6) Weight 설정

- 전이학습 사용시에 쓸 weight를 설정 할 수 있음.
- Default값은 YOLOv5s로 구조가 제일 간단하며 YOLOv5m, YOLOv5l, YOLOv5x등이 있음.



- 구조가 복잡할수록 성능이 향상될 가능성이 높지만 학습 때 더 많은 시간과 리소스가 요구됨.
- Weight 설정하는 방법

```
--weights "yolov5m.pt"
```



# 학습 파라미터 설정

## 7) 트레이닝 디바이스 선택

- 데이터셋을 학습할 때 디바이스를 선택할 수 있음. (GPU 또는 CPU)
- 디바이스 선택 하는 방법

```
--device 0(GPU) or 0,1,2,3, "cpu"등
```

## 8) 학습된 모델 이름 설정

- 학습 정보를 runs폴더 안에 저장해 줄 때 이름을 정할 수 있음.
- 기본값으로 exp1,2,...생성되는데 이후에 알아보기 힘들 수 있다
- 이름 설정 하는 방법

```
--cfg 'models/yolov5m.yaml'(예시)
```

# 학습 파라미터 설정

## 9) 학습 재시작

- 학습이 어떤 이유로 인하여 중간에 멈췄을 때 최근에 저장된 weight "last.py"(예시)부터 학습을 다시 시작할 수 있음.
- Epoch또한 멈췄던 지점부터 다시 학습이 되며 특정 weight부터 학습을 다시 시작할 수도 있음.
- 재시작 하는 방법

```
$python train.py --resume (또는 특정 경로 선택 가능 $python train.py --resume runs/test/weights/last.py)
```

## 10) 다른 이미지로 모델 실행

- '--source : detect.py'를 이용하여 다른 데이터를 가지고 모델을 실행시켜볼 수 있음.
- 사용법

```
--source ~/~/images/~~~~~.jpg
```

# 학습 파라미터 설정

## 11) 그 외

```
--line-thickness 값 : 라인 굵기  
--hide-labels true(T) or false(F) : 라벨 숨기기  
--hide-conf true or false : 설정 숨기기  
--project 이름 : 프로젝트 이름 설정 (--project / --name 함께 사용)  
--task 이름 : 테스트 결과 저장 경로 default : val  
--save-txt T or F: 라벨 결과값 txt파일로 저장  
--save-json T or F: 라벨 결과값을 coco annotation json파일 포맷으로 저장  
--save-conf T or F: txt파일 저장할 때 confidenc값도 함께 저장  
--iou-thres 값 : iou threshol값 설정  
--exist-ok T or F  
--nosave  
--worker 값 : 데이터 로더 수
```

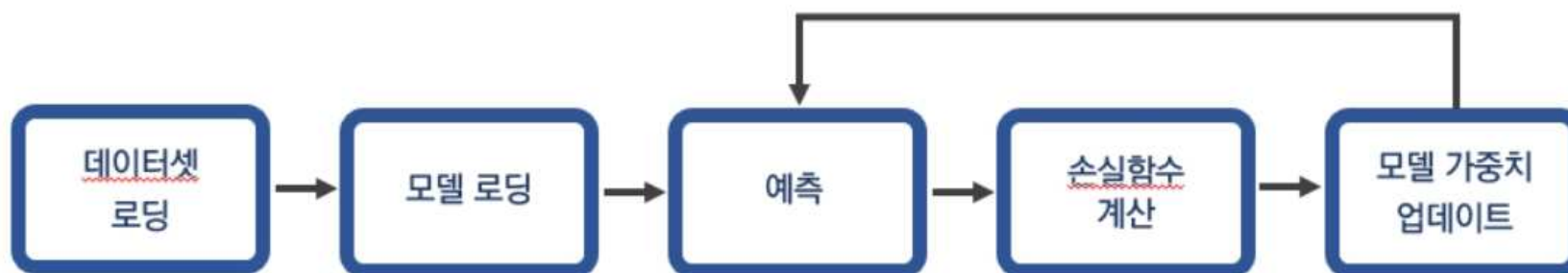
# 학습 파라미터 설정

## 12) YOLOv5 성능 높이는 팁

- ① 기본적으로 데이터 셋이 충분히 크고 라벨링이 정확하게 되어 있어야 함
- ② 백그라운드 이미지 넣기 (탐지할 물체가 없는 배경) : BG를 넣어주면 False Positives(FP)가 줄어드는 효과를 볼 수 있다. 전체 학습 데이터 셋에 0~10%정도 넣는 것을 추천
- ③ Pretrained weights 사용하기
- ④ Epoch는 300으로 시작해서 overfit이 발생하면 줄이고 발생하지 않으면 600 -> 1200으로 늘려 사용
- ⑤ 이미지 사이즈는 작은 크기의 물체가 많을 수록 높거나 원래 해상도 사이즈로 사용, Input이미지 사이즈가 학습 때 설정한 이미지 사이즈와 같아야 최고의 인퍼런스 성능을 냄
- ⑥ 기본값으로 저장된 모델 하이퍼 파라미터들은 data폴더 안에 hyp.scratch.yaml에서 찾을 수 있다. 먼저 하이퍼 파라미터로 학습하는 것을 추천, 인자값으로 -evolve를 주면 hyperparameter evolution사용 가능

# 학습/검증/추론 방법

## 1. 모델 학습 순서



1) YOLOv5 path안에 train.py를 실행하면 위와 같은 과정이 진행된다. 파라미터는 epoch 100을 넣었을 때 `!python train.py -data "data/custom_dataset.yaml" -epochs 100` 처럼 명령을 입력할 수 있다.

2) 학습이 완료되면 runs/train/exp 경로에 학습 결과가 저장된다. 학습을 반복할 경우 같은 runs/train 경로에 순서대로 exp1, 2, 3, ... 순으로 폴더가 생성되면서 학습 결과가 기록된다. (default 설정일 때)

3) 학습 결과를 다운로드 하려면 `!zip -r 압축파일명.zip /content/yolov5/runs/train/exp` (학습결과 경로) 처럼 입력한다.

# 학습/검증/추론 방법

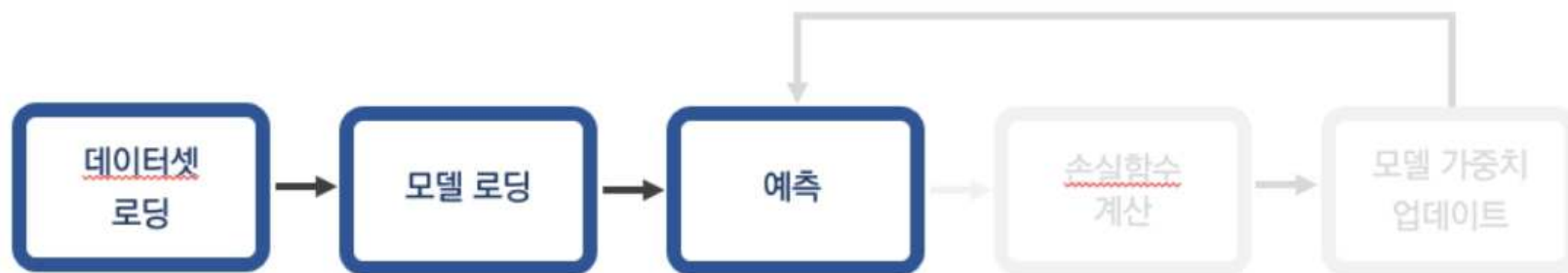
## 2. 모델 검증 순서



- 1) YOLOv5 path안에 val.py를 실행하면 위와 같은 과정이 진행된다. 모델 검증은 모델 학습 순서에서 모델 가중치 업데이트 과정이 생략된 것이다. 파라미터 값으로 데이터 경로와 모델 가중치를 사용하면  
`!python val.py -data "data/custom_dataset.yaml" -weights "content/yolov5/runs/train/exp/weights/best.pt"` 정도로 입력 할 수 있다.  
(학습한 모델 가중치는 runs/train/exp/weights/에 저장된다)
- 2) 검증이 완료되면 runs/val/exp 경로에 학습 결과가 저장된다. 학습과 마찬가지로 같은 runs/val경로에 순서대로 exp1, 2, 3, ...순으로 폴더가 생성된다.
- 3) 압축 및 다운로드도 학습과 같이 하면 된다.  
검증 exp폴더 안에는 confusion matrix, F1 curve 등 성능과 관련된 차트도 함께 저장된다.

# 학습/검증/추론 방법

## 3. 예측 과정

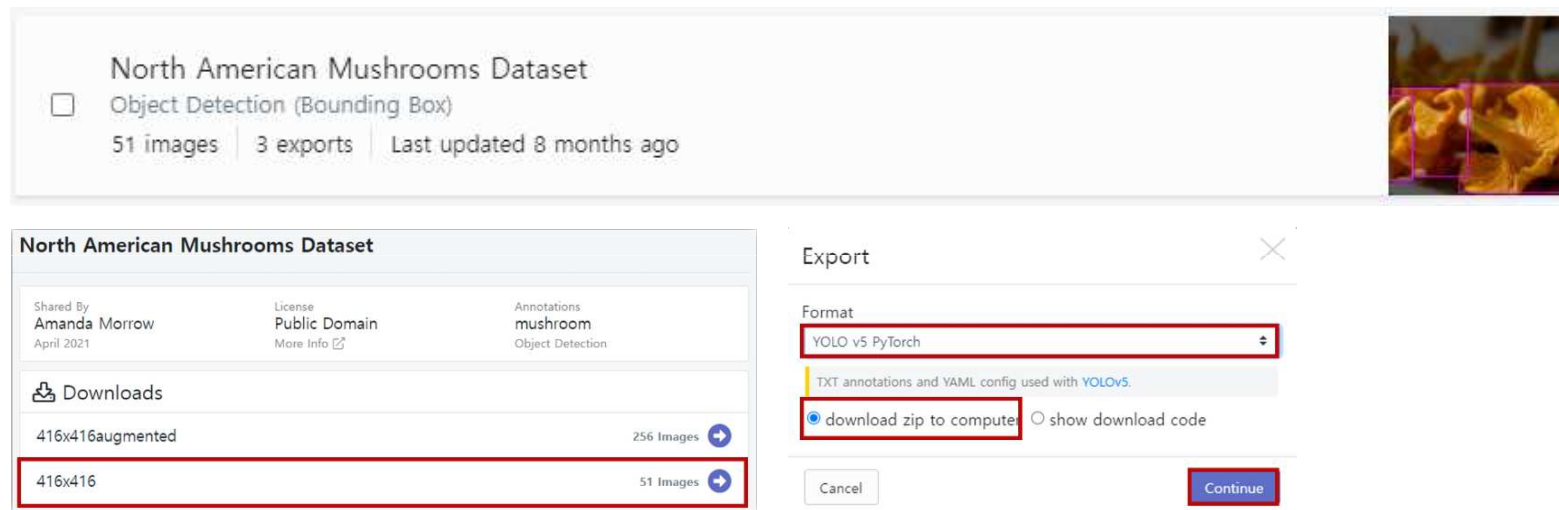


1) YOLOv5 path안에 detect.py를 실행하면 위와 같은 과정이 진행된다. 예측은 이미지 뿐만 아니라 웹캠 비디오 파일들도 실행 가능하다. -weights, --source인자를 사용하여 소스를 설정해 주면 `!python detect.py --weights "content/yolov5/runs/train/exp/weights/best.pt" --source ~/~.jpg` 처럼 입력할 수 있다.  
(jpg와 0(webcam), mp4, path/, '유튜브경로', http stream등도 가능하다.)

2) 테스트 결과는 /runs/detect/exp경로에 저장된다. (압축 및 다운로드 가능)  
폴더안에는 class와 bounding box가 표시된 detection결과 이미지가 저장되어 있다.

## 1. North American Mushrooms Dataset

- 1) 실습에 사용되는 데이터 셋은  
roboflow에서 제공되는 North American Mushrooms Dataset 다운로드



The screenshot shows the Roboflow interface for the North American Mushrooms Dataset. The dataset is listed with 51 images and 3 exports. The export options are shown in a modal window, with 'YOLO v5 PyTorch' selected as the format and 'download zip to computer' selected as the export method. The '416x416' download option is highlighted in the dataset list.

**custom\_dataset.zip**

- 2) [Colab]에 접속하고 하드웨어 가속기를 GPU로 설정  
“런타임”-”런타임 유형 변경”



The screenshot shows the '노트 설정' (Notebook Settings) dialog box in Colab. The '하드웨어 가속기' (Hardware Accelerator) is set to 'GPU'. The '백그라운드 실행' (Run in background) checkbox is unchecked. The '이 노트를 저장할 때 코드 셀 출력 생략' (Omit code cell outputs when saving this notebook) checkbox is also unchecked. The '저장' (Save) button is highlighted.



## 3) [Colab] yolov5 학습에 필요한 dataset 환경 준비

The image shows a file explorer interface for a 'yolov5' project. The left pane shows the project's root directory with folders like 'data', 'dataset', 'models', 'runs', and 'utils'. The 'dataset' folder is highlighted with a red dashed box and labeled 'New Dir'. The right pane shows the contents of the 'dataset' folder, including 'data', 'hyps', 'images', 'scripts', and several '.yaml' files. The 'custom\_dataset.yaml' file is highlighted with a red dashed box and labeled 'New File'. A red arrow points from this file to the 'custom\_dataset' folder in the right pane, with the text 'custom\_dataset.zip 압축풀기'.

```

1 path: /content/drive/MyDrive/yolov5/dataset/custom_dataset
2 train: train/images
3 val: valid/images
4 test: test/images
5
6 nc: 2
7 names: ['Cow', 'chanterelle']
  
```

## 3) 모델 학습

```
!python train.py --data "data/custom_dataset.yaml" --epochs 100 #epoch 100회
```

```
train: weights=yolov5s.pt, cfg=, data=data/custom_dataset.yaml, hyp=data/hyps/hyp.scratch-low.yaml, epochs=100, batch_size=16, imgsz=640,
github: ⚠ YOLOv5 is out of date by 4 commits. Use 'git pull' or 'git clone https://github.com/ultralytics/yolov5' to update.
YOLOv5 🚀 v6.1-186-g3356f26 Python=3.7.13 torch=1.11.0+cu113 CUDA:0 (Tesla T4, 15110MiB)
```

GPU 가속기 사용

CPU 사용

100 epochs completed in 0.054 hours.

Optimizer stripped from runs/train/exp8/weights/last.pt, 14.5MB

Optimizer stripped from runs/train/exp8/weights/best.pt, 14.5MB

Validating runs/train/exp8/weights/best.pt...

Fusing layers...

Model summary: 213 layers, 7015519 parameters, 0 gradients, 15.8 GFLOPs

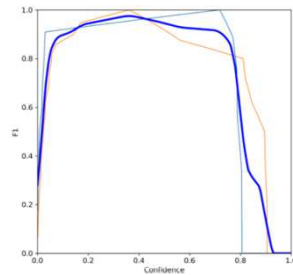
Class	Images	Labels	P	R	mAP@.5	mAP@.5: .95	100%	1/1
all	5	14	0.956	1	0.995	0.82		
CoW	5	5	0.913	1	0.995	0.861		
chanterelle	5	9	0.999	1	0.995	0.779		

Results saved to runs/train/exp8

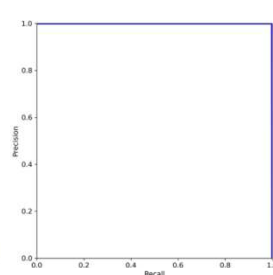
- exp8
  - weights
    - F1\_curve.png
    - PR\_curve.png
    - P\_curve.png
    - R\_curve.png
    - confusion\_matrix.png
    - events.out.tfevents.16526...
    - hyp.yaml
    - labels.jpg
    - labels\_correlogram.jpg
    - opt.yaml
    - results.csv
    - results.png
    - train\_batch0.jpg
    - train\_batch1.jpg
    - train\_batch2.jpg
    - val\_batch0\_labels.jpg
    - val\_batch0\_pred.jpg

## 4) 모델 학습 (평가지표 및 결과 - YOLOv5)

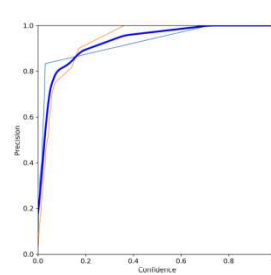
F1\_curve



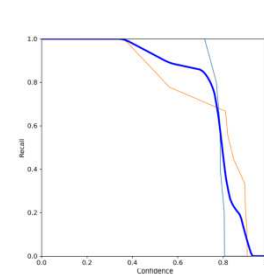
PR\_curve



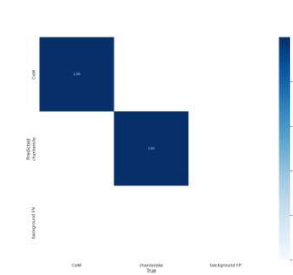
P\_curve



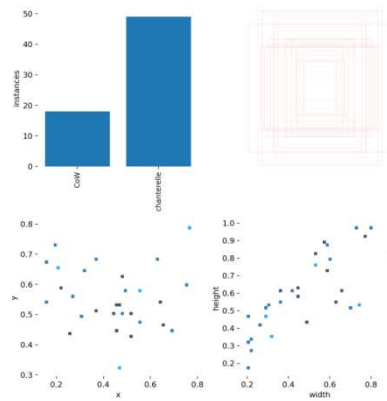
R\_curve



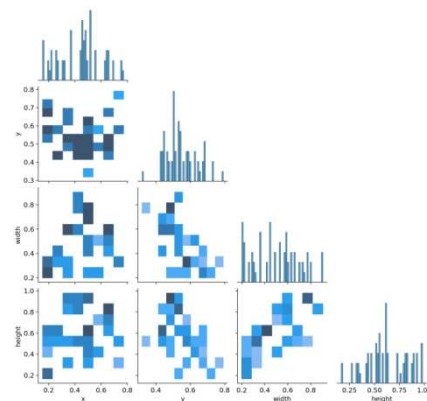
confusion\_matrix



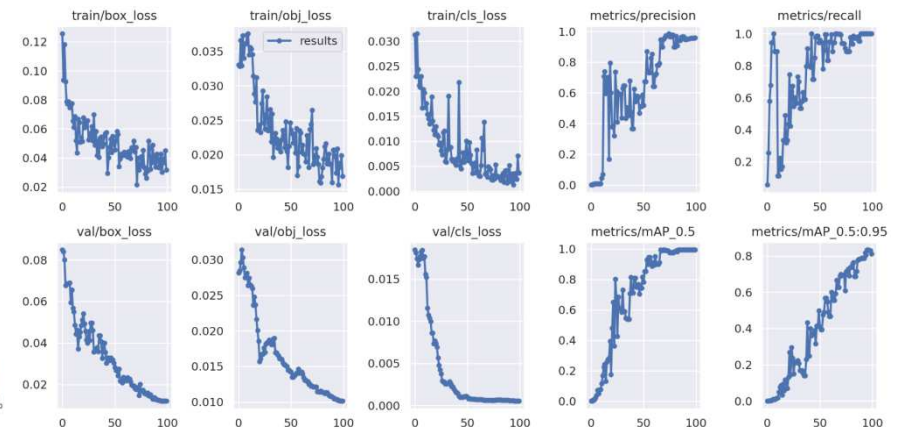
labels



labels\_correlogram



results



## 4) 모델 학습 (평가지표 및 결과 - wandb)

```
Validating runs/train/exp14/weights/best.pt...
Fusing layers...
Model summary: 213 layers, 7015519 parameters, 0 gradients, 15.8 GFLOPs
      Class      Images      Labels      P      R      mAP@.
      all         5         14      0.969      1      0.96
      CoW         5         5      0.952      1      0.95
      chanterelle  5         9      0.986      1      0.98
Results saved to runs/train/exp14
wandb: Waiting for W&B process to finish... (success).
wandb:
wandb:
wandb: Run history:
wandb: metrics/mAP_0.5
wandb: metrics/mAP_0.5:0.95
wandb: metrics/precision
wandb: metrics/recall
wandb: train/box_loss
wandb: train/cls_loss
wandb: train/obj_loss
wandb: val/box_loss
wandb: val/cls_loss
wandb: val/obj_loss
wandb: x/lr0
wandb: x/lr1
wandb: x/lr2
wandb:
wandb: Run summary:
wandb: best/epoch 98
wandb: best/mAP_0.5 0.995
wandb: best/mAP_0.5:0.95 0.82553
wandb: best/precision 0.96909
wandb: best/recall 1.0
wandb: metrics/mAP_0.5 0.995
wandb: metrics/mAP_0.5:0.95 0.82553
wandb: metrics/precision 0.96922
wandb: metrics/recall 1.0
wandb: train/box_loss 0.03321
wandb: train/cls_loss 0.00443
wandb: train/obj_loss 0.0164
wandb: val/box_loss 0.01228
wandb: val/cls_loss 0.00058
wandb: val/obj_loss 0.00837
wandb: x/lr0 0.0003
wandb: x/lr1 0.0003
wandb: x/lr2 0.0003
wandb:
wandb: Synced vital-eon-1: https://wandb.ai/dmlm/YOLOv5/runs/2wa1gp5m
wandb: Synced 5 W&B file(s), 113 media file(s), 1 artifact file(s) and 0
wandb: Find logs at: ./wandb/run-20220517\_142425-2wa1gp5m/logs
```

!pip install wandb



시각화툴

### Untitled Report

Add a description...

Go

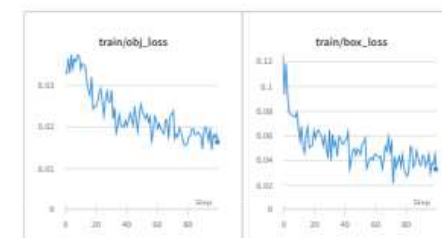
Type '/' for commands

#### Section 1



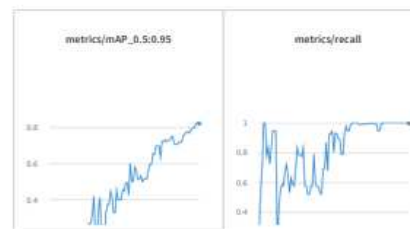
Import panel

Add panel

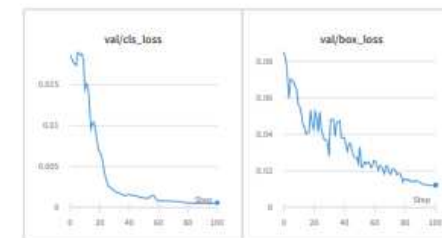
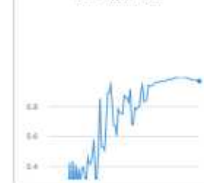


Import panel

Add panel



metrics/precision



Import panel

Add panel

## 5) 검증

```
!python val.py --data "data/custom_dataset.yaml" --weights "runs/train/exp8/weights/best.pt"
```

```
val data=data/custom_dataset.yaml weights=['runs/train/exp8/weights/best.pt'] batch_size=32, imgsz=640, conf_thres=0.001, iou_thres=0.6,
YOLOv5 v6.1-186-g3356f26 Python-3.7.13 torch-1.11.0+cu113 CUDA:0 (Tesla T4, 15110MiB)
```

Fusing layers...

Model summary: 213 layers, 7015519 parameters, 0 gradients, 15.8 GFLOPs

val: Scanning '/content/drive/MyDrive/yolov5/dataset/custom\_dataset/valid/labels.cache' images and labels... 5 found, 0 missing,

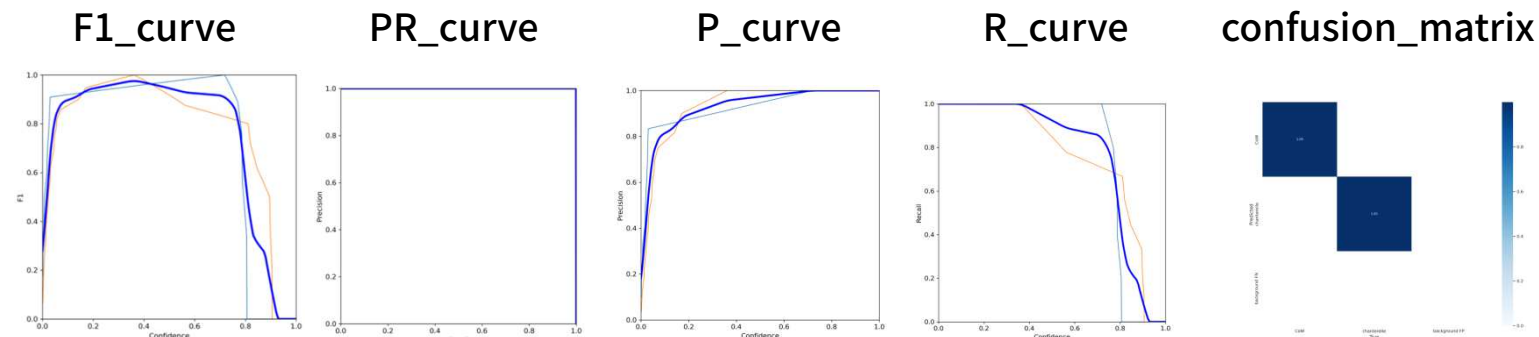
Class	Images	Labels	P	R	mAP@.5	mAP@.5:1
all	5	14	0.956	1	0.995	0.832
Colli	5	5	0.913	1	0.995	0.885
chanterelle	5	9	0.999	1	0.995	0.779

Speed: 0.4ms pre-process, 11.0ms inference, 1.5ms NMS per image at shape (32, 3, 640, 640)

Results saved to **runs/val/exp7**

exp7

- F1\_curve.png
- PR\_curve.png
- P\_curve.png
- R\_curve.png
- confusion\_matrix.png
- val\_batch0\_labels.jpg
- val\_batch0\_pred.jpg



## 6) 예측

```
!python detect.py --weights "runs/train/exp8/weights/best.pt"
--source "dataset/custom_dataset/test/images"
```

```
detect: weights=['runs/train/exp8/weights/best.pt'] source=dataset/custom_dataset/test/images, data=data/coco128.yaml, imgsz=[640, 640],
YOLOv5 v6.1-186-g3356f26 Python=3.7.13 torch=1.11.0+cu113 CUDA=0 (Tesla T4, 15110MiB)
```

Fusing layers...

Model summary: 213 layers, 7015519 parameters, 0 gradients, 15.8 GFLOPs

image 1/5 /content/drive/MyDrive/yolov5/dataset/custom\_dataset/test/images/chanterelle\_02.jpg.rf.f7a48494b7393c532f641585d99a57be.jpg: 640x640 3 chanterelles, Done. (0.013s)

image 2/5 /content/drive/MyDrive/yolov5/dataset/custom\_dataset/test/images/chanterelle\_03.jpg.rf.580f8d787af6a8050c21c065bf016f20.jpg: 640x640 2 chanterelles, Done. (0.017s)

image 3/5 /content/drive/MyDrive/yolov5/dataset/custom\_dataset/test/images/chanterelle\_03.jpg.rf.cd892d2f06d228ba20d194fc360320fc.jpg: 640x640 2 chanterelles, Done. (0.013s)

image 4/5 /content/drive/MyDrive/yolov5/dataset/custom\_dataset/test/images/chanterelle\_07.jpg.rf.6a8121422f738876d299cd11437b1855.jpg: 640x640 1 chanterelle, Done. (0.013s)

image 5/5 /content/drive/MyDrive/yolov5/dataset/custom\_dataset/test/images/chicken01.jpg.rf.6232850cea082a1ecc27f121c60ceb10.jpg: 640x640 1 Cow, Done. (0.013s)

Speed: 0.7ms preprocess, 13.8ms inference, 1.2ms NMS per image at shape (1, 3, 640, 640)

Results saved to runs/detect/exp5

exp5

chanterelle\_02.jpg.rf.f7a48494b7393c532f641585d99a57be.jpg

chanterelle\_03.jpg.rf.580f8d787af6a8050c21c065bf016f20.jpg

chanterelle\_03.jpg.rf.cd892d2f06d228ba20d194fc360320fc.jpg

chanterelle\_07.jpg.rf.6a8121422f738876d299cd11437b1855.jpg

chicken01.jpg.rf.6232850cea082a1ecc27f121c60ceb10.jpg

**Q&A**

**감사합니다**