

# Yolov5 학습가이드

## with Colab

## 목차

I	개요 .....	4
1.	문서 목적 .....	4
2.	History .....	5
II	준비 .....	6
1.	구성 .....	6
2.	학습 이미지 라벨링 .....	7
2.1	이미지 준비 .....	7
2.2	도구 준비 .....	7
2.3	도구 설정 .....	7
III	COLAB 환경구성 .....	10
1.	Colab 로그인 .....	10
2.	학습 예제 파일 적용 .....	10
3.	Colab GPU 연결 .....	10
IV	YOLOV5 설치 및 학습 .....	11
1.	설치 .....	11
2.	준비 .....	11
2.1	폴더 생성 .....	11
2.2	업로드 .....	11
3.	모델 학습 .....	12
4.	모델 학습 결과 검증 .....	13
5.	모델 변환 .....	14

### 그림 목차

[그림 II-1] 라벨링 방법 .....	8
[그림 II-2] DATA.YAML 수정 .....	9
[그림 III-1] 제공된 예제파일 업로드 .....	10
[그림 III-2] COLAB GPU 연결.....	10
[그림 IV-1] GIT을 이용한 YOLOV5 설치 .....	11
[그림 IV-2] COLAB 저장소 오픈 .....	11
[그림 IV-3] 이미지 파일을 관리하기 위한 폴더 생성 .....	11
[그림 IV-4] YOLOV5 모델 학습 .....	12
[그림 IV-5] YOLOV5 모델 학습 결과 .....	12
[그림 IV-6] TENSORBOARD를 이용하여 세부 내용 확인 .....	13
[그림 IV-7] 학습된 모델 결과 테스트 .....	13
[그림 IV-8] YOLOV5 모델을 ONNX로 내보내기 .....	14

# I 개요

---

## 1. 문서 목적

본 문서는 Colab 환경을 기반으로 ZAiV에서 구동하기 위한 YOLOv5 모델을 학습하기 위한 가이드입니다.

2. History

Version	날짜	내용
1.0	2023.10.24.	Initial Release
1.1	2024.01.24	Build up contents

## II 준비

---

### 1. 구성

가이드와 함께 제공된 아래 파일들을 준비합니다.

- Colab 학습을 위한 Jupyter 파일: yolov5\_learning\_example.ipynb
- 학습이미지 라벨링 도구: DarkLabel2.4.zip
- 데이터 구성 설정파일: data.yaml

## 2. 학습 이미지 라벨링

### 2.1 이미지 준비

학습에 사용할 이미지를 준비합니다.

---

*이미지는 실제 추론환경과 동일한 환경(구도, 색상, 사용카메라, resolution, bitrate등)으로 획득해야합니다.  
예를 들어, 같은 사람의 이미지라도, 상공(ex. 드론)에서 촬영한 것을 통해 지상에서 사용하는 경우 구도가  
다르므로 학습 후 추론 시 인식률이 현저히 떨어집니다.*

---

### 2.2 도구 준비

- DarkLabel2.4.zip 파일을 압축 해제합니다.

### 2.3 도구 설정

#### 2.3.1 열기

- darklabel.yml 파일을 엽니다.

#### 2.3.2 학습 객체 지정

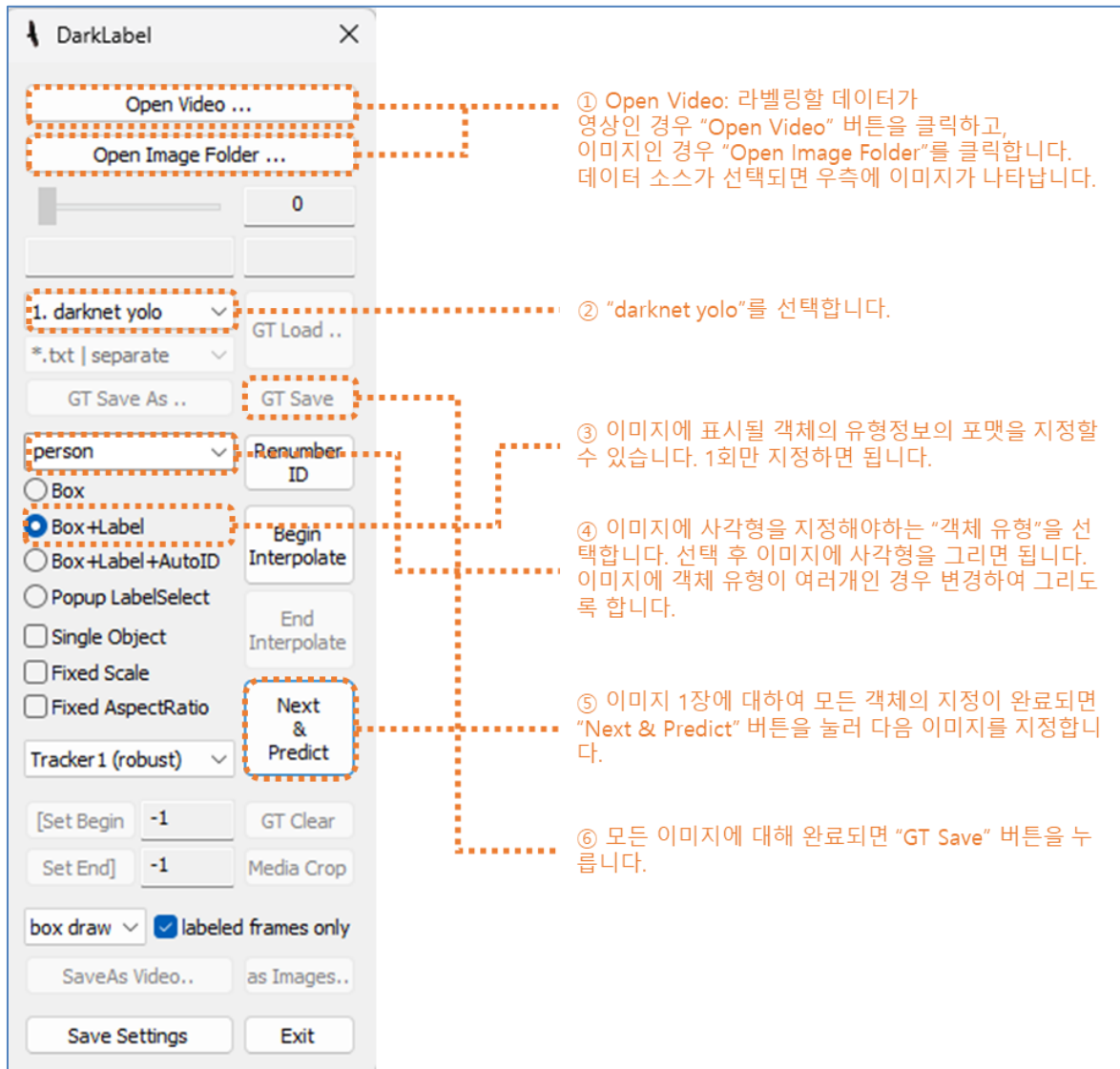
- 14번째 줄의 `my_classes1: ["class1", "class2"]`을 찾은 후 `class1`, `class2` 대신 사용할 이름들로 수정 후 저장합니다. (예시: "Can", "Paper", "Pet", ...)

#### 2.3.3 실행

- "DarkLabel.exe"를 실행합니다. ("매개 변수가 틀립니다." 메시지는 무시하셔도 됩니다.)

## 2.3.4 라벨링

- 아래 라벨링 방법에 따라 준비된 이미지들의 라벨링을 실시합니다.



The image shows the DarkLabel software interface with several key components highlighted by orange dashed boxes and numbered annotations:

- ① Open Video / Open Image Folder ...**: 라벨링할 데이터가 영상인 경우 "Open Video" 버튼을 클릭하고, 이미지인 경우 "Open Image Folder"를 클릭합니다. 데이터 소스가 선택되면 우측에 이미지가 나타납니다.
- ② "darknet yolo"**: 모델 유형을 "1. darknet yolo"로 선택합니다.
- ③ person**: 이미지에서 표시될 객체의 유형정보의 포맷을 지정할 수 있습니다. 1회만 지정하면 됩니다.
- ④ Box+Label**: 이미지에서 사각형을 지정해야 하는 "객체 유형"을 선택합니다. 선택 후 이미지에 사각형을 그리면 됩니다. 이미지에 객체 유형이 여러개인 경우 변경하여 그리도록 합니다.
- ⑤ Next & Predict**: 이미지 1장에 대하여 모든 객체의 지정이 완료되면 "Next & Predict" 버튼을 눌러 다음 이미지를 지정합니다.
- ⑥ GT Save**: 모든 이미지에 대해 완료되면 "GT Save" 버튼을 누릅니다.

[그림 II-1] 라벨링 방법



### 2.3.5 학습 준비

- 학습 시 사용할 “data.yaml”(가이드와 함께 제공되었습니다.) 파일을 열어 수정합니다.
  1. name은 객체(class)들의 이름을 적습니다. (라벨링할 때 사용한 이름과 순서를 동일하게 수정합니다.)
  2. train은 train image들이 있는 경로를 지정합니다. (colab 기본경로:/content)
  3. val은 val image들이 있는 경로를 지정합니다. (colab 기본경로:/content)

```
names: # Classes name
- class1
- class2

train: /content/yolov5/images # train images 경로
val: /content/yolov5/images # val images 경로
```

[그림 II-2] data.yaml 수정

### III Colab 환경구성

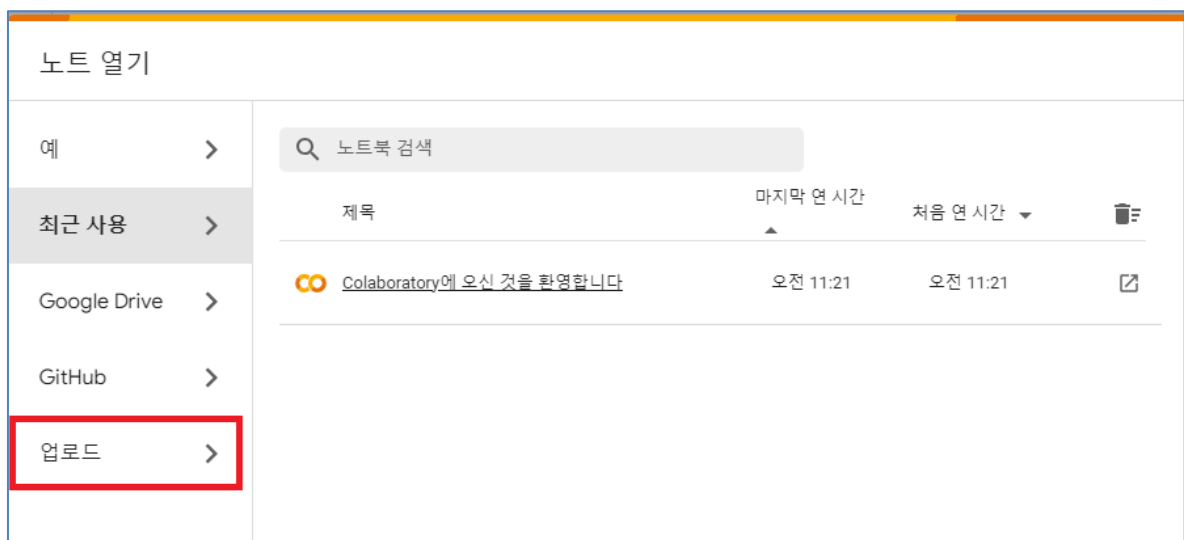
Colab 학습 환경 구성을 위한 구성방법입니다.

#### 1. Colab 로그인

- <https://colab.research.google.com/>로 접속합니다
- 우측상단의 로그인 버튼을 눌러 구글계정으로 로그인 합니다.

#### 2. 학습 예제 파일 적용

- 가이드와 함께 제공된 `yolov5_learning_example.ipynb` 파일을 준비합니다.
- 로그인 후 나타난 팝업창에서 업로드를 클릭합니다

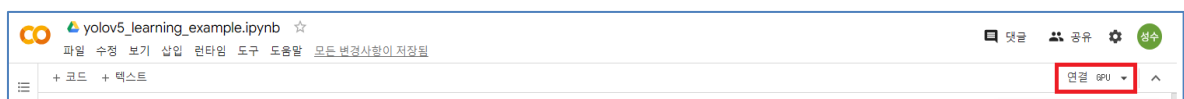


[그림 III-1] 제공된 예제파일 업로드

- 둘러보기를 클릭하여 가이드와 함께 제공된 `yolov5_learning_example.ipynb` 파일을 선택하여 업로드합니다.

#### 3. Colab GPU 연결

- 우측상단의 “연결 GPU”를 선택하여 Colab에 GPU를 연결합니다.



[그림 III-2] Colab GPU 연결

## IV YoloV5 설치 및 학습

### 1. 설치

- yolov5 깃 서버에 접속하여 clone합니다. Clone 후 yolov5의 프로젝트가 colab에 다운로드 됩니다.
- 이 후 yolov5폴더로 이동 후 Requirements.txt파일을 통해 필요한 파이썬 라이브러리들을 일괄 설치합니다.

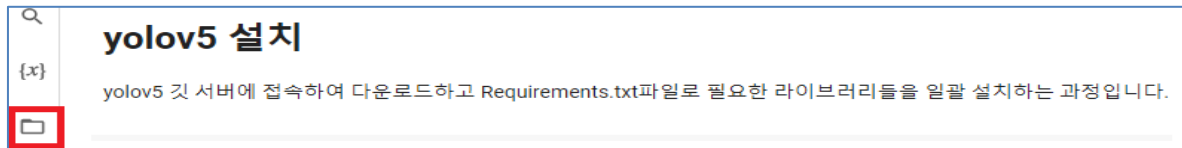
```
[ ] 1 #clone YOLOv5 and
    2 !git clone https://github.com/ultralytics/yolov5 # clone repo
    3 %cd yolov5
    4 %pip install -qr requirements.txt # install dependencies
```

[그림 IV-1] git을 이용한 yolov5 설치

### 2. 준비

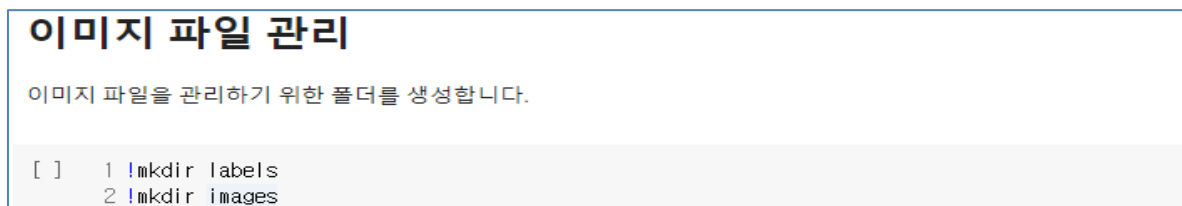
#### 2.1 폴더 생성

- 좌측의 폴더 모양을 클릭하여 Colab 저장소를 열고 위에서 설치한 yolov5 폴더를 열면 설치된 yolov5 파일들이 확인됩니다.



[그림 IV-2] Colab 저장소 오픈

- 라벨 폴더와 이미지폴더를 생성합니다.



[그림 IV-3] 이미지 파일을 관리하기 위한 폴더 생성

#### 2.2 업로드

- labels 폴더에는 라벨링된 .txt 파일들을 업로드합니다.
- images 폴더에는 학습할 이미지들을 업로드합니다.
- 앞서 수정한 data.yaml 파일을 yolov5 폴더에 업로드합니다.

## 3. 모델 학습

- yolov5 학습은 아래의 명령을 통해 동작합니다.

### yolov5 모델 학습 시작

```
[ ] 1 #필요 라이브러리 임포트하기
    2 import torch
    3 import os
    4 from IPython.display import Image, clear_output # to display images

[ ] 1 #모델 학습하기
    2 !python train.py --img 512 --batch 16 --epochs 500 --data /content/yolov5/data.yaml --weights /content/yolov5/yolov5n.pt --cache
```

[그림 IV-4] yolov5 모델 학습

- train.py 의 속성들은 아래와 같습니다.
  1. img=이미지 사이즈
  2. batch=학습 이미지 단위
  3. epochs=학습 반복 횟수
  4. data=데이터 셋 정보가 있는 yaml 파일 경로
  5. weights=가중치 파일
  6. cache=캐시메모리 사용 여부
- 학습이 완료되면 /content/yolov5/runs/train/exp 경로에 Loss가 가장 적었던 best모델(best.pt)이 저장됩니다.

Epoch	GPU_mem	box_loss	obj_loss	cls_loss	Instances	Size	
499/499	1.24G	0.0102	0.005515	0.0001176	6	512: 100% 35/35 [00:05<00:00, 5.96it/s]	
	Class	Images	Instances	P	R	mAP50	mAP50-95: 100% 18/18 [00:04<00:00, 3.66it/s]
	all	548	732	1	1	0.995	0.93

500 epochs completed in 1.520 hours.  
 Optimizer stripped from runs/train/exp/weights/last.pt, 3.8MB  
 Optimizer stripped from runs/train/exp/weights/best.pt, 3.8MB

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

Fusing layers...

Model summary: 157 layers, 1761871 parameters, 0 gradients, 4.1 GFLOPs

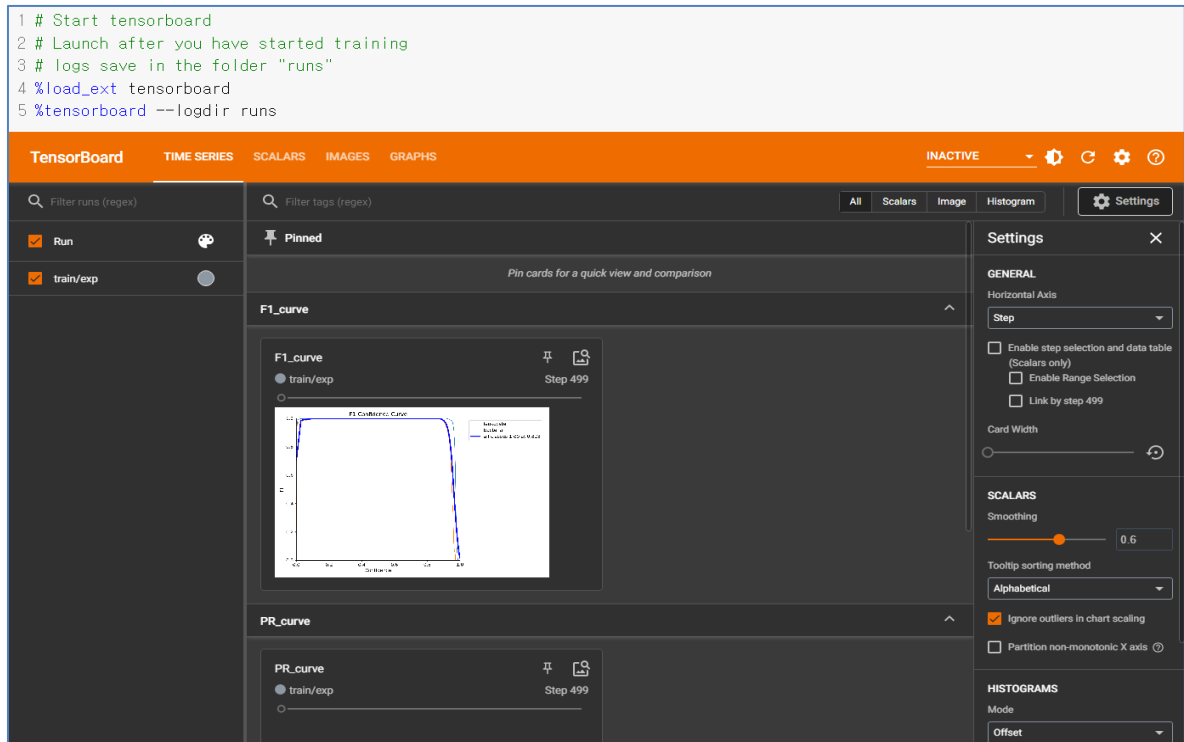
Class	Images	Instances	P	R	mAP50	mAP50-95	
all	548	732	0.999	1	0.995	0.93	100% 18/18 [00:06<00:00, 2.97it/s]
leukocyte	548	653	1	1	0.995	0.969	
bacteria	548	79	0.999	1	0.995	0.89	

Results saved to **runs/train/exp**

[그림 IV-5] yolov5 모델 학습 결과

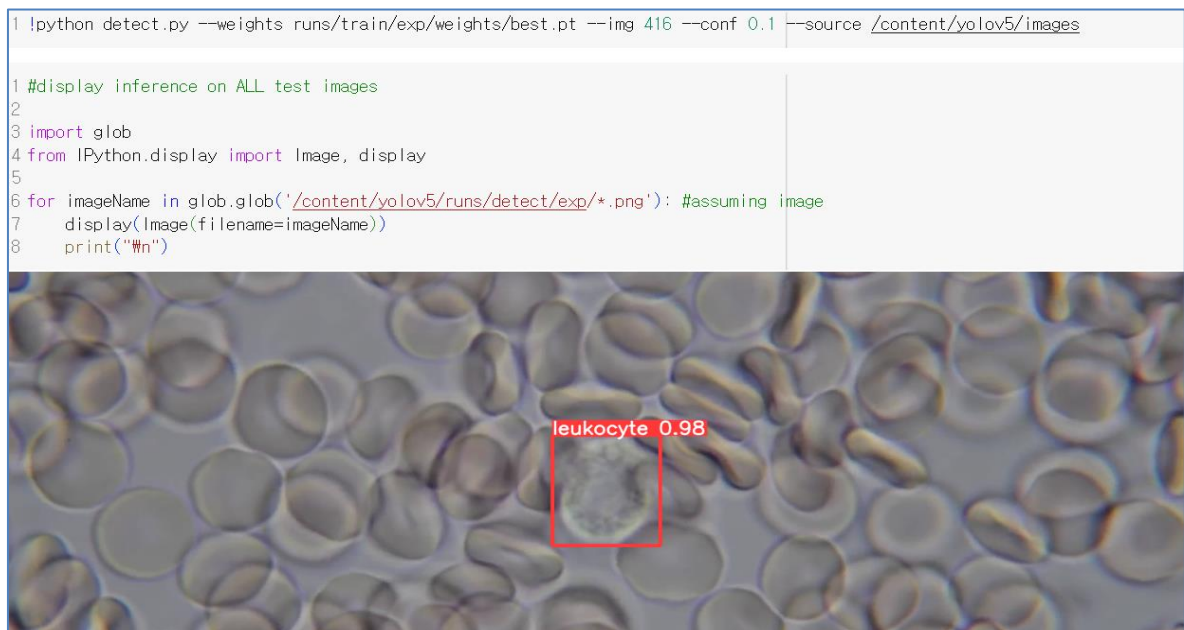
## 4. 모델 학습 결과 검증

- tensorboard를 이용하여 모델이 학습된 세부 내용들을 확인할 수 있습니다.



[그림 IV-6] tensorboard를 이용하여 세부 내용 확인

- detect.py를 학습된 모델로 추론 결과를 테스트할 수 있습니다.



[그림 IV-7] 학습된 모델 결과 테스트

## 5. 모델 변환

- export.py를 이용해 onnx파일로 저장해주면 커스텀 데이터 셋으로 모델 학습이 완료되었습니다.
- onnx파일은 /content/yolov5/runs/train/exp/weights경로에 저장됩니다. (best.onnx)
- Export된 onnx파일은 ZAiV Board에 맞게 변환 시 사용됩니다.

onnx 파싱

```

[ ] | python export.py --img 512 --weights '/content/yolov5/runs/train/exp/weights/best.pt' --include torchscript onnx

export: data=data/coco128.yaml, weights=['/content/yolov5/runs/train/exp/weights/best.pt'], imgsz=[512], batch_size=1, dev
YOLov5 v7.0-227-ge4df1ec Python-3.10.12 torch-2.0.1+cu118 CPU

Fusing layers...
Model summary: 157 layers, 1761871 parameters, 0 gradients, 4.1 GFLOPs

PyTorch: starting from /content/yolov5/runs/train/exp/weights/best.pt with output shape (1, 16128, 7) (3.6 MB)

TorchScript: starting export with torch 2.0.1+cu118...
TorchScript: export success 2.9s, saved as /content/yolov5/runs/train/exp/weights/best.torchscript (7.1 MB)

ONNX: starting export with onnx 1.14.1...
===== Diagnostic Run torch.onnx.export version 2.0.1+cu118 =====
verbose: False, log level: Level.ERROR
===== 0 NONE 0 NOTE 0 WARNING 0 ERROR =====

ONNX: export success 0.7s, saved as /content/yolov5/runs/train/exp/weights/best.onnx (7.0 MB)

Export complete (4.3s)
Results saved to /content/yolov5/runs/train/exp/weights
Detect: python detect.py --weights /content/yolov5/runs/train/exp/weights/best.onnx
Validate: python val.py --weights /content/yolov5/runs/train/exp/weights/best.onnx
PyTorch Hub: model = torch.hub.load('ultralytics/yolov5', 'custom', '/content/yolov5/runs/train/exp/weights/best.onnx')
Visualize: https://netron.app

```

[그림 IV-8] yolov5 모델을 onnx로 내보내기

- Yolov5 학습은 완료되었습니다.
- 이후에 DFC 가이드를 참고하여 hef 파일로 컴파일하시면 됩니다.