ZAiV

Yolov5 학습가이드

with Anaconda

https://www.with-us.kr

WITHUS

목차

그림 목차

I 개요

1. 문서 목적

본 문서는 Anaconda 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.

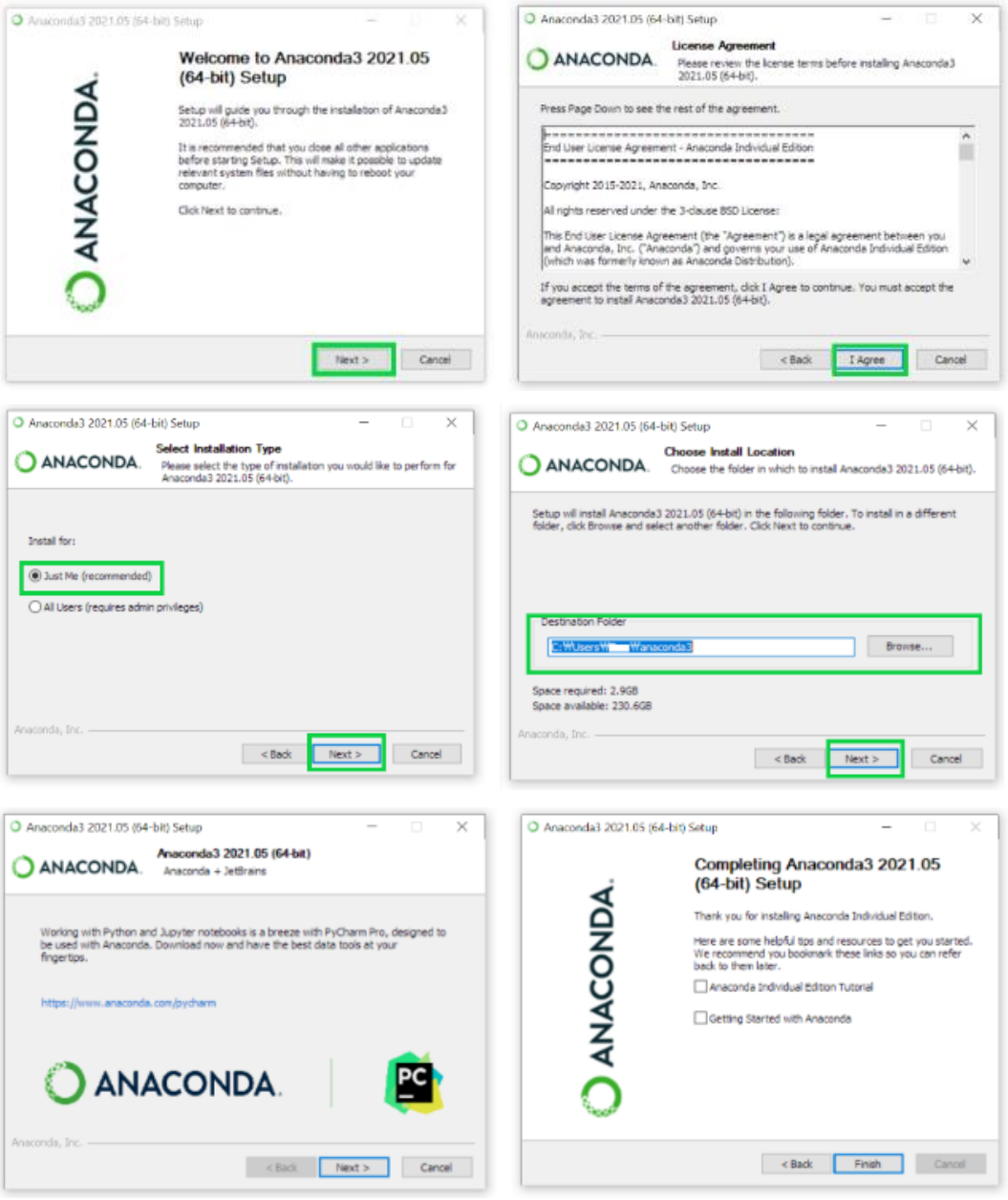
- **Anaconda 환경에서 Yolov5 학습을 위한 Jupyter 파일**: yolov5_relearning.ipynb
- **학습이미지 라벨링 도구**: DarkLabel2.4.zip
- **데이터 구성 설정파일**: data.yaml

2. Anaconda 설치

아나콘다는 파이썬을 사용하는 유저들에게 편의성을 제공하기위해 여러 도구(패키지)들을 모아둔 오픈소스 배포판입니다.

https://www.anaconda.com/download - 아나콘다 공식 다운로드 사이트에 접속하여 아나콘다 설치파일을 다운로드합니다.

설치된 exe파일을 실행하면 아래와 같은 순서로 진행됩니다.



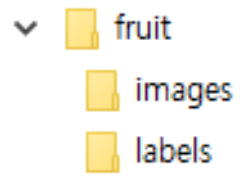
[그림 II-1] Anaconda 설치 방법

3. 학습 이미지 라벨링

3.1 이미지 준비

학습에 사용할 이미지를 준비합니다.

폴더 구조는 데이터셋 이름 하위폴더로 images, labels 폴더 두개를 만들어서 구성해줍니다.



[그림 II-2] 데이터셋 폴더 구조

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.

3.2 도구 준비

- DarkLabel2.4.zip 파일을 압축 해제합니다.

3.3 도구 설정

3.3.1 열기

- darklabel.yml 파일을 엽니다.

3.3.2 학습 객체 지정

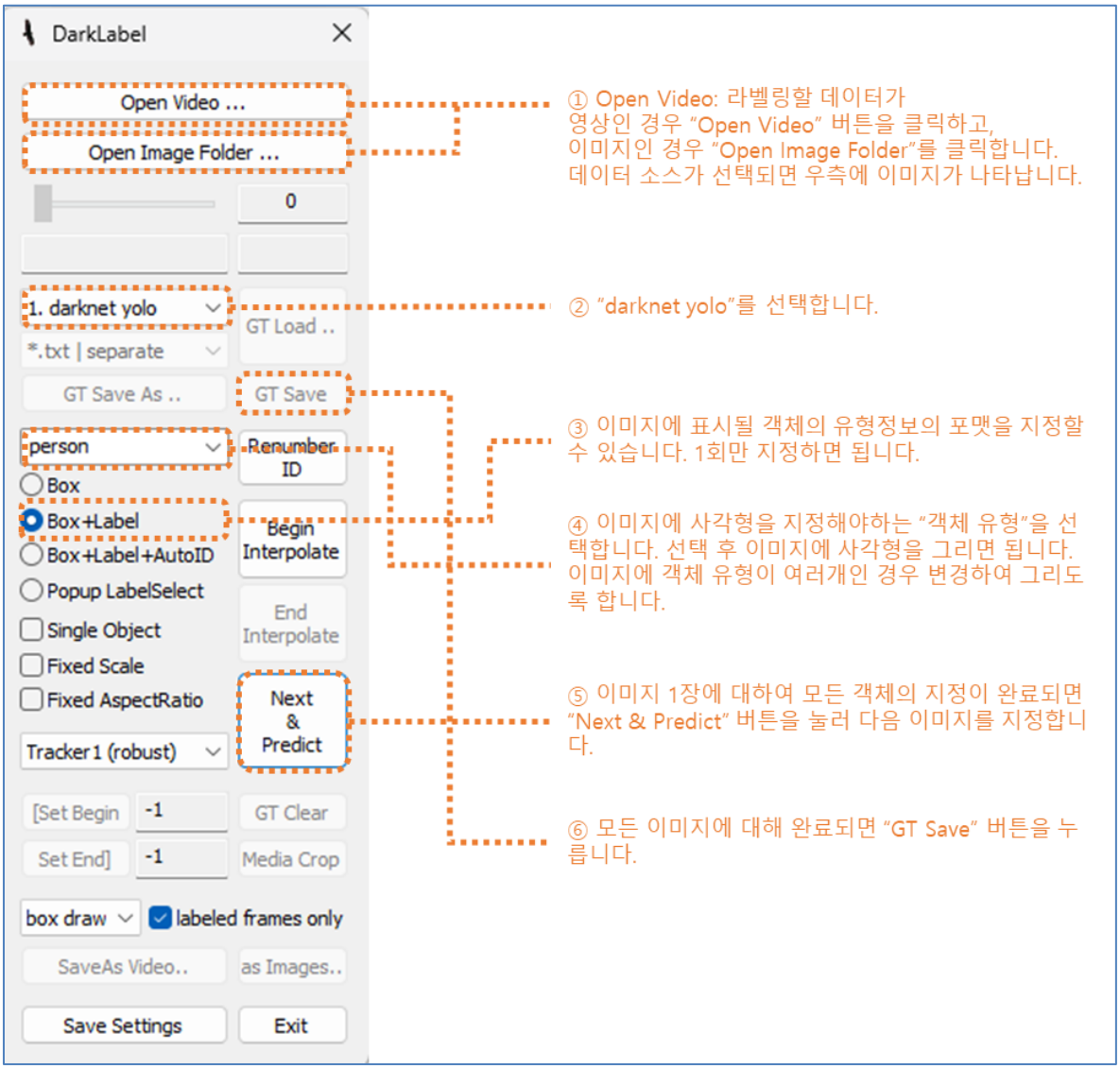
- 14번째 줄의 **my_classes1: ["class1", "class2"]**을 찾은 후 class1, class2 대신 사용할 이름들로 수정 후 저장합니다. (예시: “Can”, “Paper”, “Pet”, …)

3.3.3 실행

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

3.3.4 라벨링

- 아래 라벨링 방법에 따라 준비된 이미지들의 라벨링을 실시합니다.



[그림 II-3] 라벨링 방법

III Anaconda 환경구성

Anaconda 학습 환경 구성을 위한 구성방법입니다.

1. Conda 환경 생성

시작메뉴에서 Anaconda Prompt 라는 프로그램을 찾아 실행하고 아래 명령어를 순서대로 입력합니다.

```
conda create --name yolov5 python=3.9
conda activate yolov5
```

그러면 "(base) C:\Users**$username**"에서 "(yolov5) C:\Users**$username**"으로 변경됩니다.

그 상태에서 원하는 작업 폴더 경로로 이동합니다.

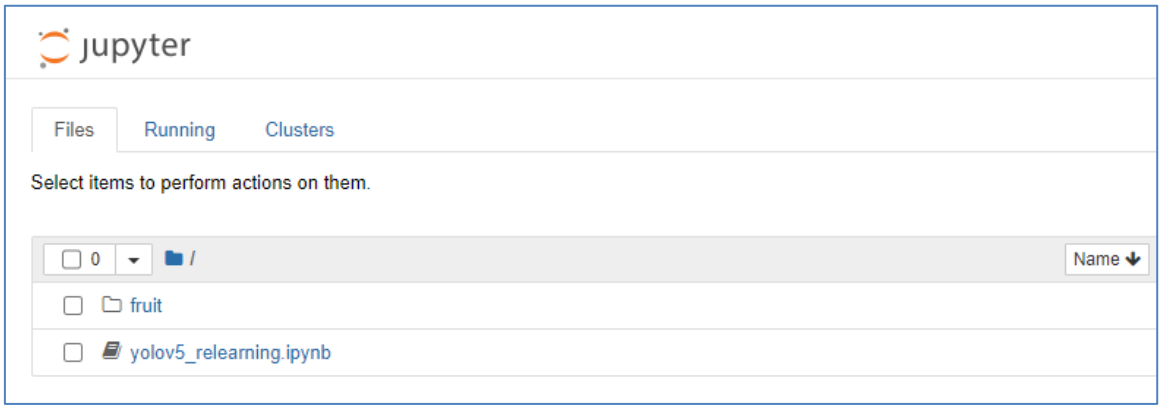
> 드라이브 이동 방법 => D: **or** C: **or** F:
>
> 파일 탐색기에서 작업 폴더로 이동 후 파일 경로를 복사해서 붙여넣기하고 엔터하면 이동됩니다.

2. 학습 예제 파일 적용

- 가이드와 함께 제공된 **yolov5_relearning.ipynb** 파일을 작업폴더에 준비합니다.
- 라벨링한 데이터 폴더를 같은 작업 폴더에 옮겨줍니다.

3. Jupyter notebook 실행

- Anaconda Prompt에서 jupyter notebook을 실행해주면 아래와 같은 화면이 출력됩니다.



[그림 III-1] Jupyter notebook 실행

- yolov5_relearning.ipynb 파일을 더블클릭하여 열어줍니다.

IV YoloV5 설치 및 학습

1. 설치

- yolov5 깃 서버에 접속하여 clone합니다. Clone 후 yolov5의 프로젝트가 작업 폴더에 다운로드 되고 yolov5폴더로 이동 하여 Requirements.txt파일을 통해 필요한 파이썬 라이브러리들을 일괄 설치 합니다.

첫 설치

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

#clone YOLOv5 and
%cd yolov5
#install dependencies
!pip install -r requirements.txt
!pip install comet_ml tensorboard onnx scikit-learn
```

[그림 IV-1] yolov5 설치

- 이후 필요한 패키지를 import 합니다.

```
import torch
import os
# to display images
from IPython.display import Image, clear_output

print(f"Setup complete. Using torch {torch.__version__} ({torch.cuda.get_device_properties(0).name if
```

[그림 IV-2] 필요한 패키지 import

2. 준비

2.1 폴더 생성

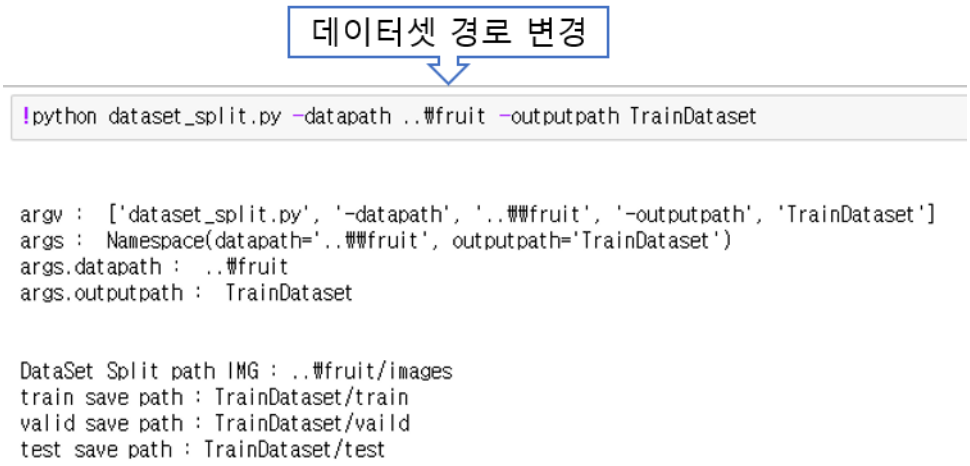
- 데이터셋을 [학습 : train, 검증 : vaild, 확인 : test] 용도별로 나눠서 저장하기위해 폴더를 만듭니다.

```
%mkdir TrainDataset\\train\\images
%mkdir TrainDataset\\train\\labels
%mkdir TrainDataset\\vaild\\images
%mkdir TrainDataset\\vaild\\labels
%mkdir TrainDataset\\test\\images
```

[그림 IV-3] 데이터 셋 폴더 생성

2.2 업로드

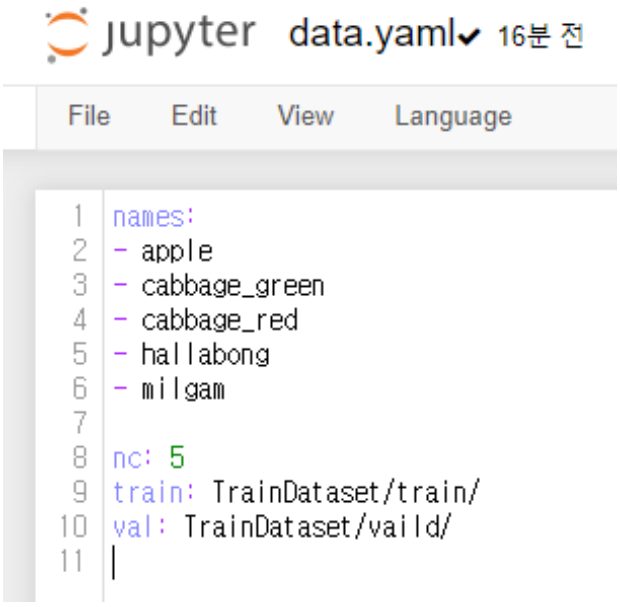
- 가이드와 제공된 dataset_split.py 파일을 yolov5 폴더로 이동하고 다음 코드를 실행합니다.



[그림 IV-4] 라벨링 데이터 분배하기

- 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. (기본경로: TrainDataset/train)
  3. val은 val image들이 있는 경로를 지정합니다. (기본경로: TrainDataset/vaild)

예시)



[그림 IV-5] data.yaml 파일 예시

3. 모델 학습

- yolov5 학습은 아래의 명령을 통해 동작합니다.

```
!python train.py --img 512 --batch 16 --epochs 30 --data data.yaml --weights yolov5n.pt
```

```
cabbage_red         11          3       0.707          1       0.995       0.863
  hallabong         11          2       0.782          1       0.995       0.995
     milgam         11          3        0.53          1       0.995       0.764
Results saved to runs\train\exp
COMET INFO: ----------------------------------------------------------------------------------------------------
COMET INFO: Comet.ml OfflineExperiment Summary
COMET INFO: ----------------------------------------------------------------------------------------------------
COMET INFO:   Data:
COMET INFO:     display_summary_level : 1
COMET INFO:     url                   : [OfflineExperiment will get URL after upload]
COMET INFO:   Metrics [count] (min, max):
COMET INFO:     loss [15]                 : (0.7919555902481079, 3.731377363204956)
COMET INFO:     metrics/mAP_0.5 [60]      : (0.0, 0.995)
COMET INFO:     metrics/mAP_0.5:0.95 [60] : (0.0, 0.8160235451805046)
COMET INFO:     metrics/precision [60]    : (0.0, 0.9686769246129266)
COMET INFO:     metrics/recall [60]       : (0.0, 1.0)
```

[그림 IV-6] 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 yolov5/runs/train/exp 경로에 Loss가 가장 적었던 best모델(best.pt)이 저장됩니다.

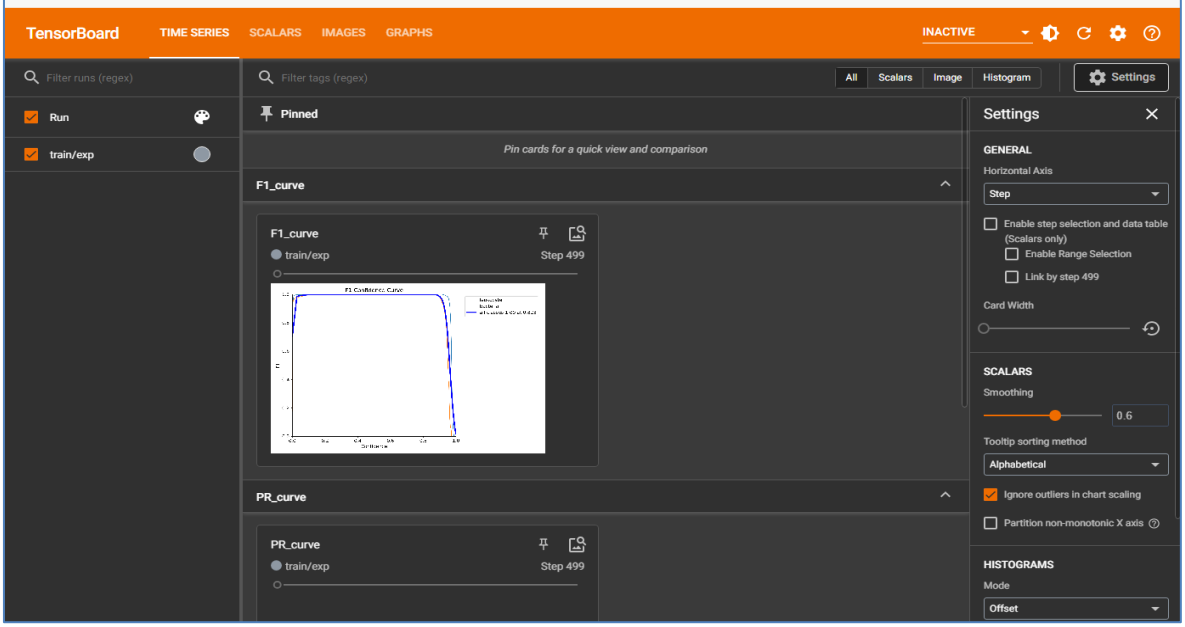
```
     Epoch    GPU_mem   box_loss   obj_loss   cls_loss  Instances       Size
   499/499      1.24G     0.0102   0.005515  0.0001176          6        512: 100% 35/35 [00:05<00:00,  5.96it/s]
                Class     Images  Instances          P          R      mAP50   mAP50-95: 100% 18/18 [00:04<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: 100% 18/18 [00:06<00:00,  2.97it/s]
                  all        548        732      0.999          1      0.995       0.93
            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-7] yolov5 모델 학습 결과

4. 모델 학습 결과 검증

- tensorboard를 이용하여 모델이 학습된 세부 내용들을 확인할 수 있습니다.

```
# Start tensorboard
# Launch after you have started training
# logs save in the folder "runs"
%load_ext tensorboard
%tensorboard --logdir runs
```



[그림 IV-8] tensorboard를 이용하여 세부 내용 확인

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

```
!python detect.py --weights runs/train/exp/weights/best.pt --img 512 --conf 0.1 --source TrainDataset₩₩test₩₩images
```

```
#display inference on ALL test images

import glob
from IPython.display import Image, display

for imageName in glob.glob('runs/detect/exp2/*.png'): #assuming png
    display(Image(filename=imageName))
    print("₩n")
```

[그림 IV-9] 학습된 모델 결과 테스트

5. 모델 변환

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

onnx 변환

```
!python export.py --opset 15 --weights runs/train/exp/weights/best.pt --include torchscript onnx

D:₩01_hailo₩Conda₩yolov5₩export.py:59: DeprecationWarning:
Pyarrow will become a required dependency of pandas in the next major release of pandas (pandas 3.0),
(to allow more performant data types, such as the Arrow string type, and better interoperability with other libraries)
but was not found to be installed on your system.
If this would cause problems for you,
please provide us feedback at https://github.com/pandas-dev/pandas/issues/54466

  import pandas as pd
export: data=D:₩01_hailo₩Conda₩yolov5₩data₩coco128.yaml, weights=['runs/train/exp/weights/best.pt'], imgsz=[640, 640], ba
ice=cpu, half=False, inplace=False, keras=False, optimize=False, int8=False, per_tensor=False, dynamic=False, simplify=Fal
rbose=False, workspace=4, nms=False, agnostic_nms=False, topk_per_class=100, topk_all=100, iou_thres=0.45, conf_thres=0.25
hscript', 'onnx']
YOLOv5  v7.0-278-g050c72c Python-3.9.18 torch-2.1.2+cpu CPU

Fusing layers...
Model summary: 157 layers, 1765930 parameters, 0 gradients, 4.1 GFLOPs

PyTorch: starting from runs₩train₩exp₩weights₩best.pt with output shape (1, 25200, 10) (3.7 MB)

TorchScript: starting export with torch 2.1.2+cpu...
TorchScript: export success  1.5s, saved as runs₩train₩exp₩weights₩best.torchscript (7.2 MB)

ONNX: starting export with onnx 1.15.0...
ONNX: export success  0.3s, saved as runs₩train₩exp₩weights₩best.onnx (7.2 MB)

Export complete (2.0s)
Results saved to D:₩01_hailo₩Conda₩yolov5₩runs₩train₩exp₩weights
Detect:          python detect.py --weights runs₩train₩exp₩weights₩best.onnx
Validate:        python val.py --weights runs₩train₩exp₩weights₩best.onnx
PyTorch Hub:     model = torch.hub.load('ultralytics/yolov5', 'custom', 'runs₩train₩exp₩weights₩best.onnx')
Visualize:       https://netron.app
```

[그림 IV-10] yolov5 모델을 onnx로 내보내기

- Yolov5 학습이 완료되었습니다.
- 이 후 DFC를 통해 hef 파일로 컴파일하시면 됩니다.