Colab Yolov5 Learning Guide

(v1.2)

목차

그림 목차

I 개요

1. 문서 목적

본 문서는 Colab 환경을 기반으로 Yolov5 모델을 학습하기 위한 가이드입니다.

2. History

Version	날짜	내용
1.0	2023.10.24.	Initial Release
1.1	2023.10.25	sample_labeling_images 추가
1.2	2024.01.02	Darklabel 툴 사용 방법 수정

II 준비

1. 구성

가이드와 함께 제공된 아래 파일리스트를 확인합니다.

- **yolov5_learning_example.ipynb**
- **DarkLabel2.4.zip**
- **data.yaml**
- **yolov5n.pt**
- **sample_labeling_images.zip (샘플 이미지, 라벨링 데이터)**

2. 데이터 셋 이미지 라벨링

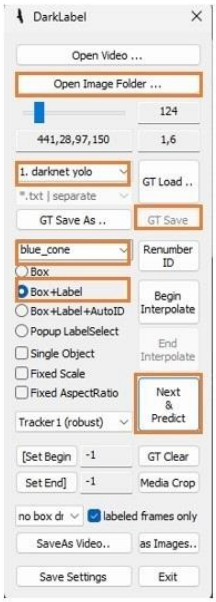
- 가이드와 함께 제공된 **DarkLabel2.4.zip** 파일의 압축을 해제합니다.
- 커스텀 데이터 셋의 class를 사용하기위해 압축을 해제시킨후 darklabel.yml 파일을 열고 14번째 줄의 **my_classes1: ["person", "vehicle", "bicycle", "motorbike", "animal", "tree", "building"] 라인**을 찾은 후 **복사 & 붙여넣기**를 한 후에 my_classes2로 바꾸고 대괄호 안에 내용은 자신이 라벨링을 할 class 이름으로 수정하여 저장합니다.
  class개수만큼 입력해주면 됩니다. 아래가 예시입니다.

```
my_classes1: ["person", "vehicle", "bicycle", "motorbike", "animal", "tree", "building"]
(ctrl+c, ctrl+v)
my_classes2: ["apple", "banana"]
```

  그리고 아래로 내리면 format1 블록이 있습니다.
  해당 블록에서 classes_set을 커스텀 데이터 셋으로 설정한 이름으로 변경해줍니다.

```
format1:    # darknet yolo (predefined format]
  fixed_filetype: 1                  # if specified a
  data_fmt: [classid, ncx, ncy, nw, nh]
  gt_file_ext: "txt"                  # if not specif
  gt_merged: 0                      # if not specifie
  delimiter: " "                      # if not specif
  classes_set: "my_classes2"      # if not specified,
  name: "darknet yolo"            # if not specified,
```

- DarkLabel.exe를 실행합니다. (“매개 변수가 틀립니다.” 라는 메시지는 무시하시면 됩니다.)
- 그 후 아래의 사진을 참고하여 라벨링 작업을 해주세요.



1. Open Image Folder 클릭하여 이미지가 있는 폴더 선택, 그럼 옆에 이미지가 뜸.

2. "1. darknet yolo" 선택

3. 이제 옆의 이미지에 해당 클래스명 선택 후 그림
   [참고] "box+label" 선택.(한번만 선택하면 됨.이미지로 as Image 버튼 클릭할 때 박스+레이블 채로 나갈 수 있어서 포맷을 지정하는건데, 사실상 화면을 구분하기 위한 용도)

4. 하나 이미지 다 그리면 "Next&Predict" 클릭.

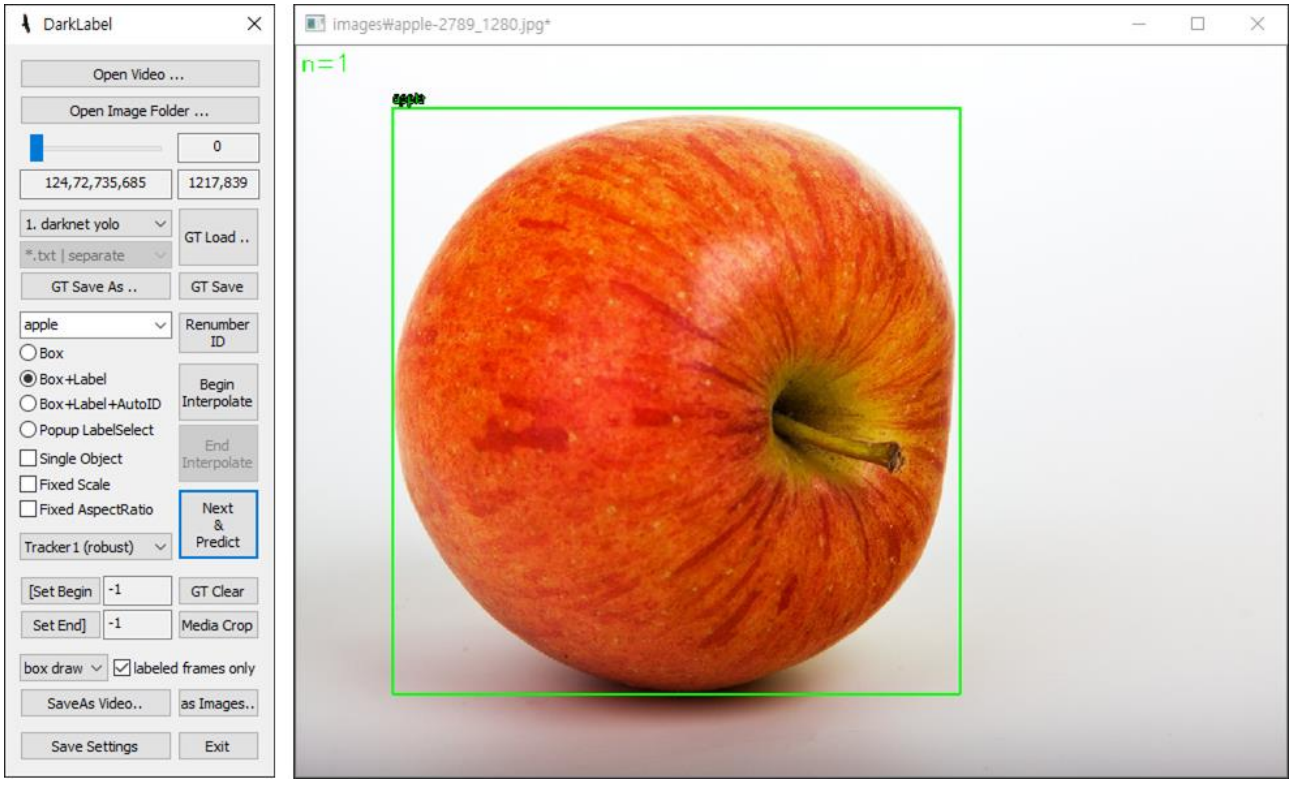
5. GT Save를 통해 라벨링데이터 출력.

단축키
- shift + leftclick -> 이동 및 조정
- shift + right click -> 삭제
- shift + double click -> 수정
- double click -> 해당 클래스 모두 선택
- space bar -> 다음 이미지

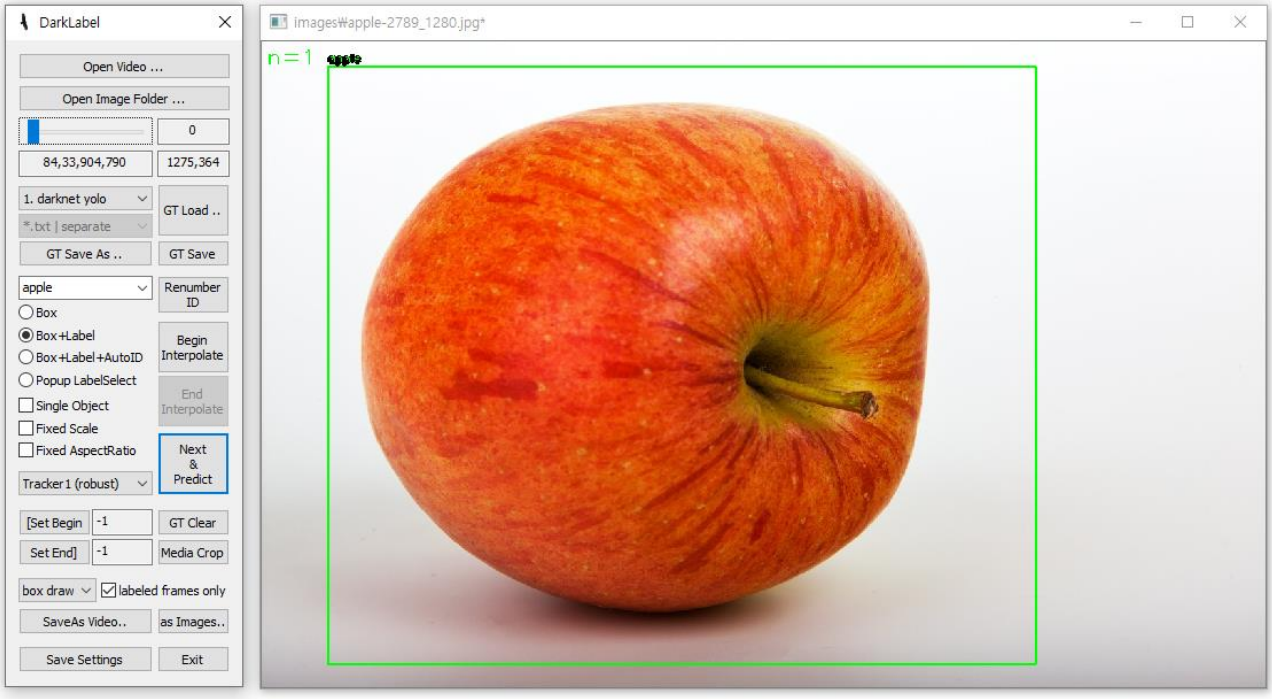
[그림 II-1] 라벨링 툴 사용법

▪ 라벨링 시 주의사항

라벨링 하려는 객체와 라벨링된 사각형 사이의 틈이 거의 없어야 학습할 때 제대로된 학습이 됩니다.

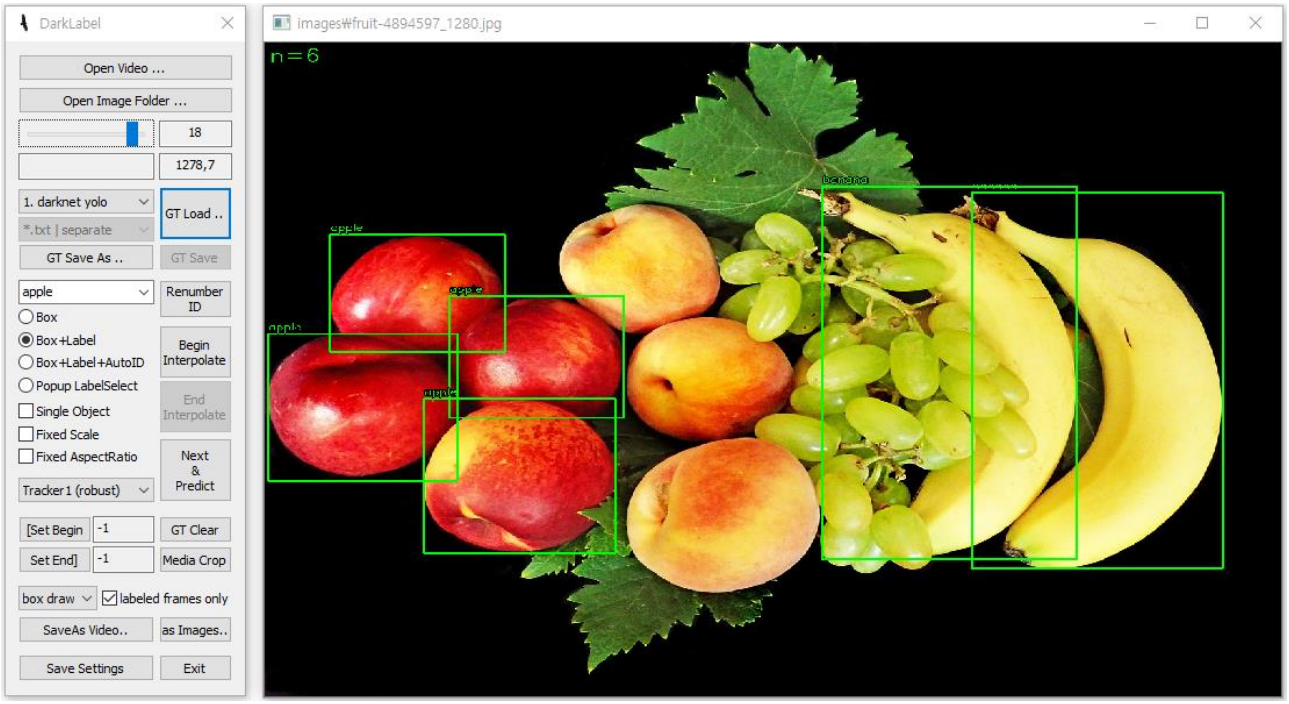


[그림 II-2] 라벨링을 잘한 예시1

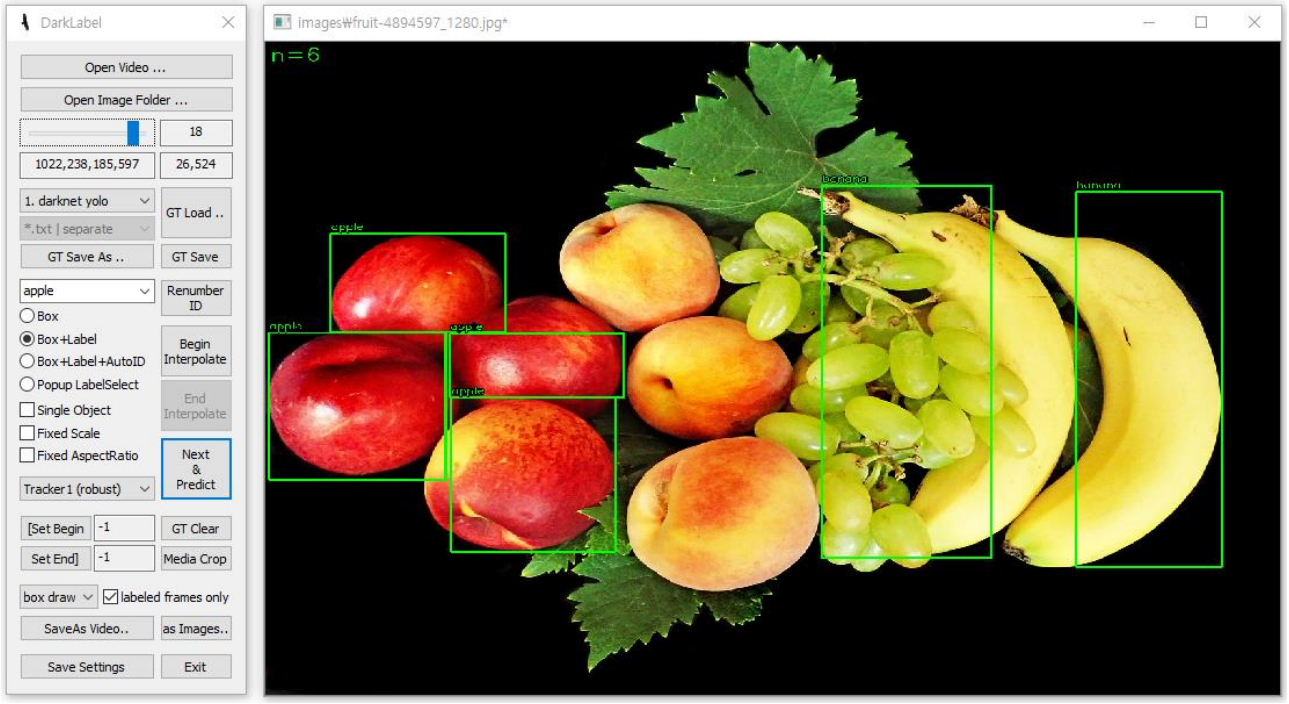


[그림 II-3] 라벨링 수정이 필요한 예시1

여러 개의 객체를 라벨링 할때 객체끼리 겹쳐진 부분도 같이 라벨링을 해야 합니다. 라벨링을 하려는 객체가 겹쳐졌다고 겹쳐진 부분에서 라벨링을 멈추지 말고 라벨링을 할 객체가 있다고 예상되는 부분까지 라벨링이 되어야 합니다.

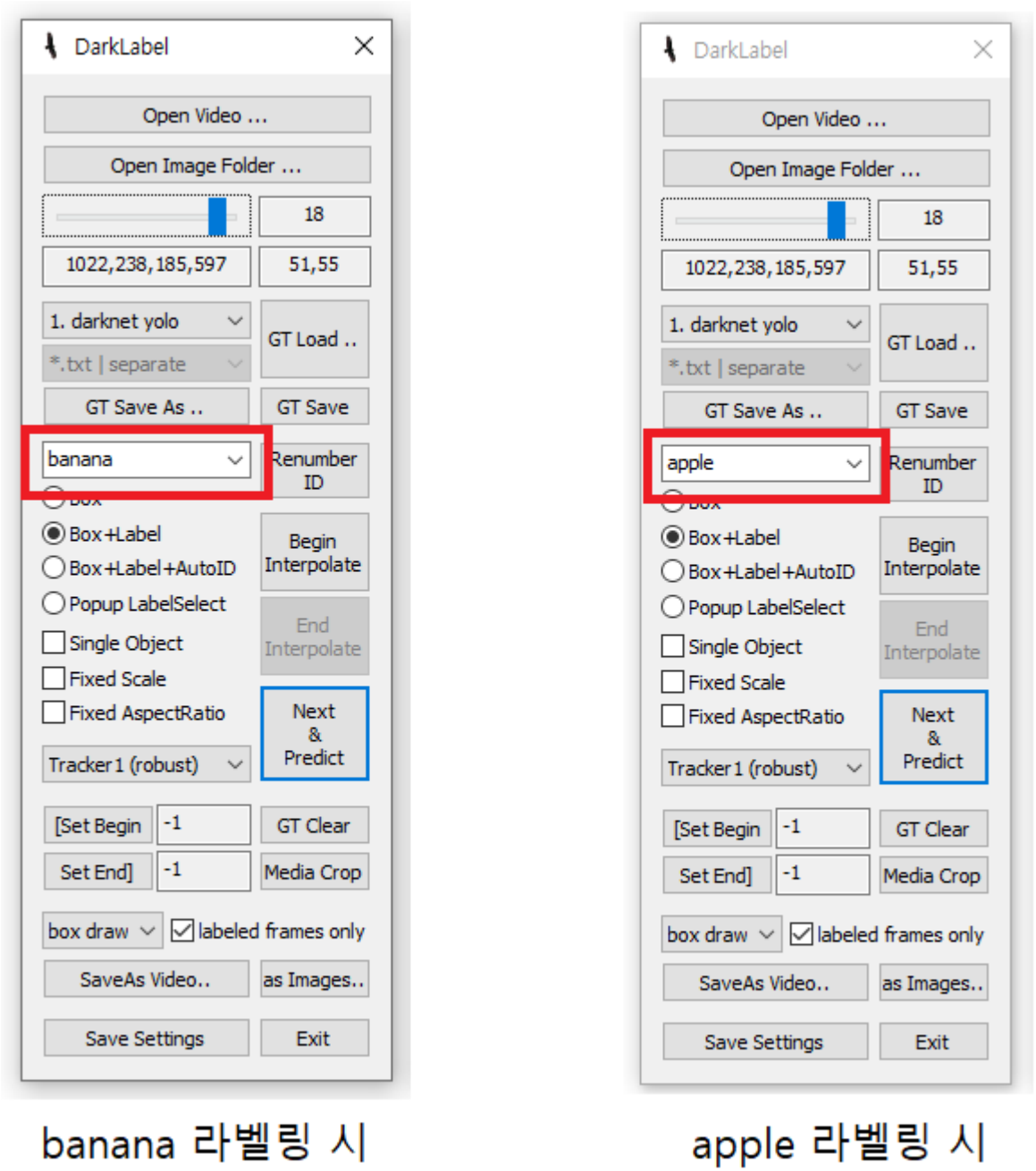


[그림 II-4] 라벨링을 잘한 예시2



[그림 II-5] 라벨링 수정이 필요한 예시2

라벨링을 할 때 라벨링을 하려는 객체의 class 이름이 바뀌면 라벨링 툴의 선택된 class 이름도 변경해야 합니다.



[그림 II-6] 라벨링 하려는 객체의 이름이 바뀐 경우

- 가이드와 함께 제공된 data.yaml 파일을 열어 아래 내용을 참고하여 수정합니다.

    1. name은 class들의 이름을 적습니다.(라벨링할 때 사용한 이름과 순서를 맞춰 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-7] data.yaml 수정

III Colab 학습 환경구성

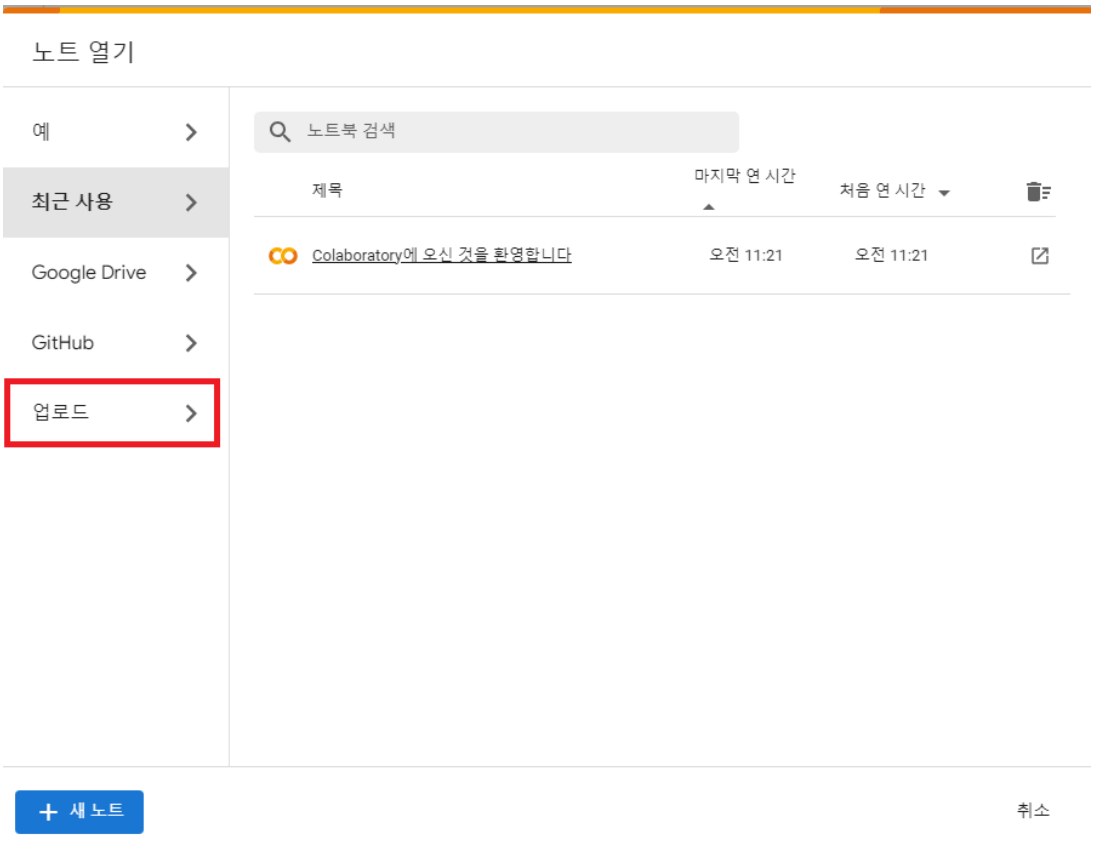
Colab 학습 환경 구성을 위한 구성방법입니다.

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

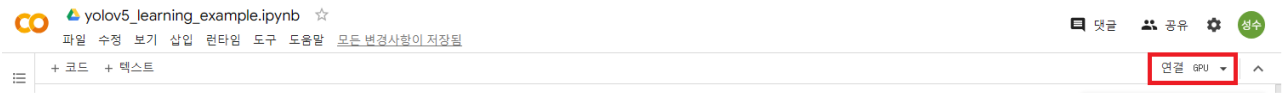


[그림 III-1] 제공된 예제 파일 업로드

- 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 / 학습

1. yolov5 설치

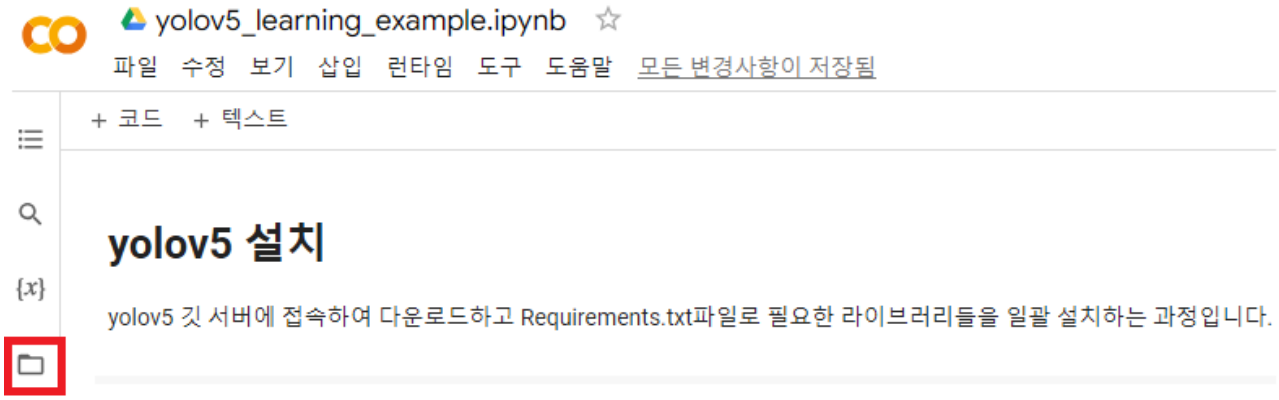
yolov5 깃 서버에 접속하여 다운로드하고 Requirements.txt파일로 필요한 라이브러리들을 일괄 설치하는 과정입니다.

```
[ ]  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. 학습에 필요한 파일들 추가

- 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 만든 후 yolov5 폴더를 선택하여 labels 폴더에는 라벨링된 .txt 파일들을 넣고 images 폴더에는 이미지들을 넣습니다.

이미지 파일 관리

이미지 파일을 관리하기 위한 폴더를 생성합니다.

```
[ ]  1 !mkdir labels
     2 !mkdir images
```

라벨 폴더와 이미지 폴더를 만든 후 yolov5 폴더를 선택하여 각각의 폴더에 이미지를 넣습니다.

[그림 IV-3] 이미지 파일을 관리하기 위한 폴더 생성

- 가이드와 함께 제공된 yolov5n.pt 파일과 수정한 data.yaml 파일도 yolov5 폴더에 넣습니다.

3. yolov5 모델 학습

- yolov5 학습은 아래의 커맨드로 동작한다.

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 **img=이미지 사이즈**, **batch=학습 이미지 단위**, **epochs=학습 반복 횟수**, **data=데이터 셋 정보가 있는 yaml 파일 경로**, **weights=가중치 파일**, **cache=캐시메모리 사용 여부**입니다.
- 학습이 완료되면 /content/yolov5/runs/train/exp 경로에 Loss가 가장 적었던 best모델(best.pt)이 저장됩니다.

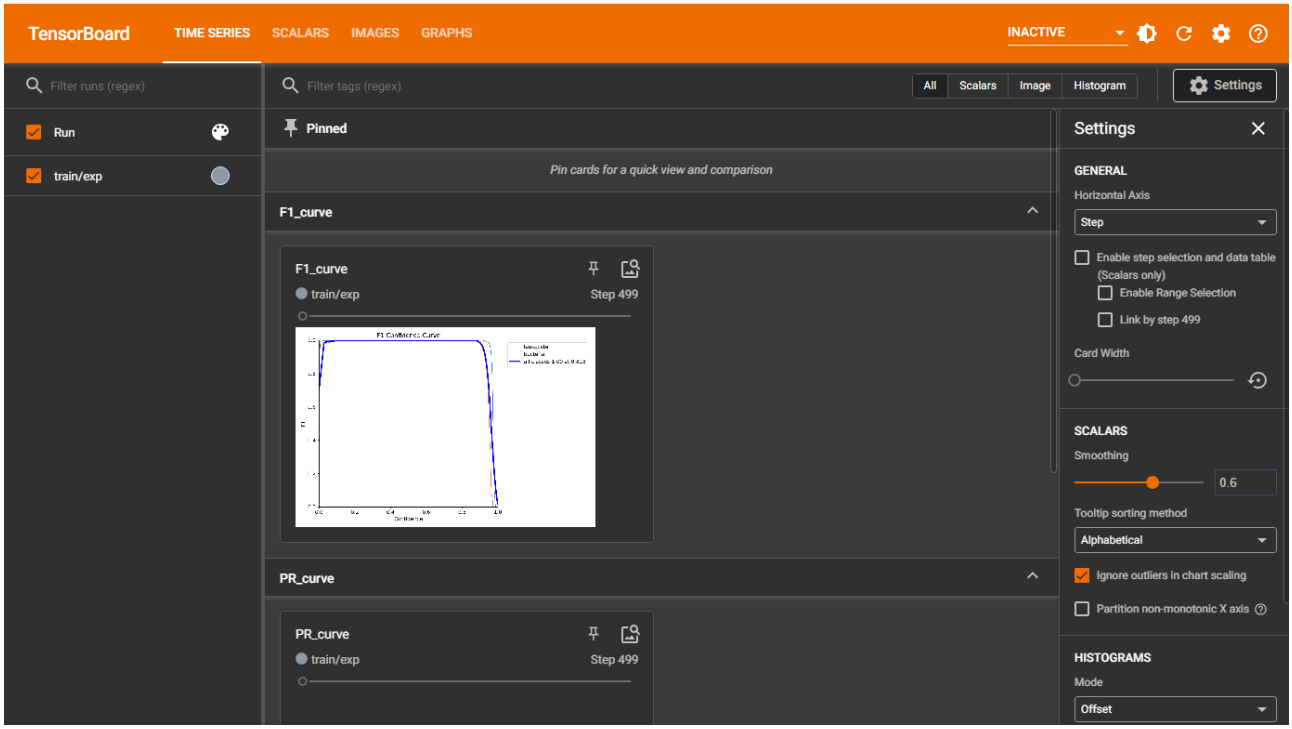
```
      Epoch    GPU_mem   box_loss   obj_loss   cls_loss  Instances       Size
  496/499      1.24G   0.009949   0.005446  5.668e-05         10        512: 100% 35/35 [00:06<00:00,  5.73it/s]
                 Class     Images  Instances          P          R      mAP50   mAP50-95: 100% 18/18 [00:03<00:00,  4.75it/s]
                   all        548        732      0.999          1      0.995      0.928

      Epoch    GPU_mem   box_loss   obj_loss   cls_loss  Instances       Size
  497/499      1.24G    0.01067   0.006002  8.373e-05         16        512: 100% 35/35 [00:07<00:00,  4.84it/s]
                 Class     Images  Instances          P          R      mAP50   mAP50-95: 100% 18/18 [00:03<00:00,  4.80it/s]
                   all        548        732      0.999          1      0.995      0.929

      Epoch    GPU_mem   box_loss   obj_loss   cls_loss  Instances       Size
  498/499      1.24G    0.01041   0.005724  5.402e-05         13        512: 100% 35/35 [00:07<00:00,  4.68it/s]
                 Class     Images  Instances          P          R      mAP50   mAP50-95: 100% 18/18 [00:03<00:00,  4.59it/s]
                   all        548        732      0.999          1      0.995      0.931

      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-5] yolov5 모델 학습 결과

4. yolov5 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
```

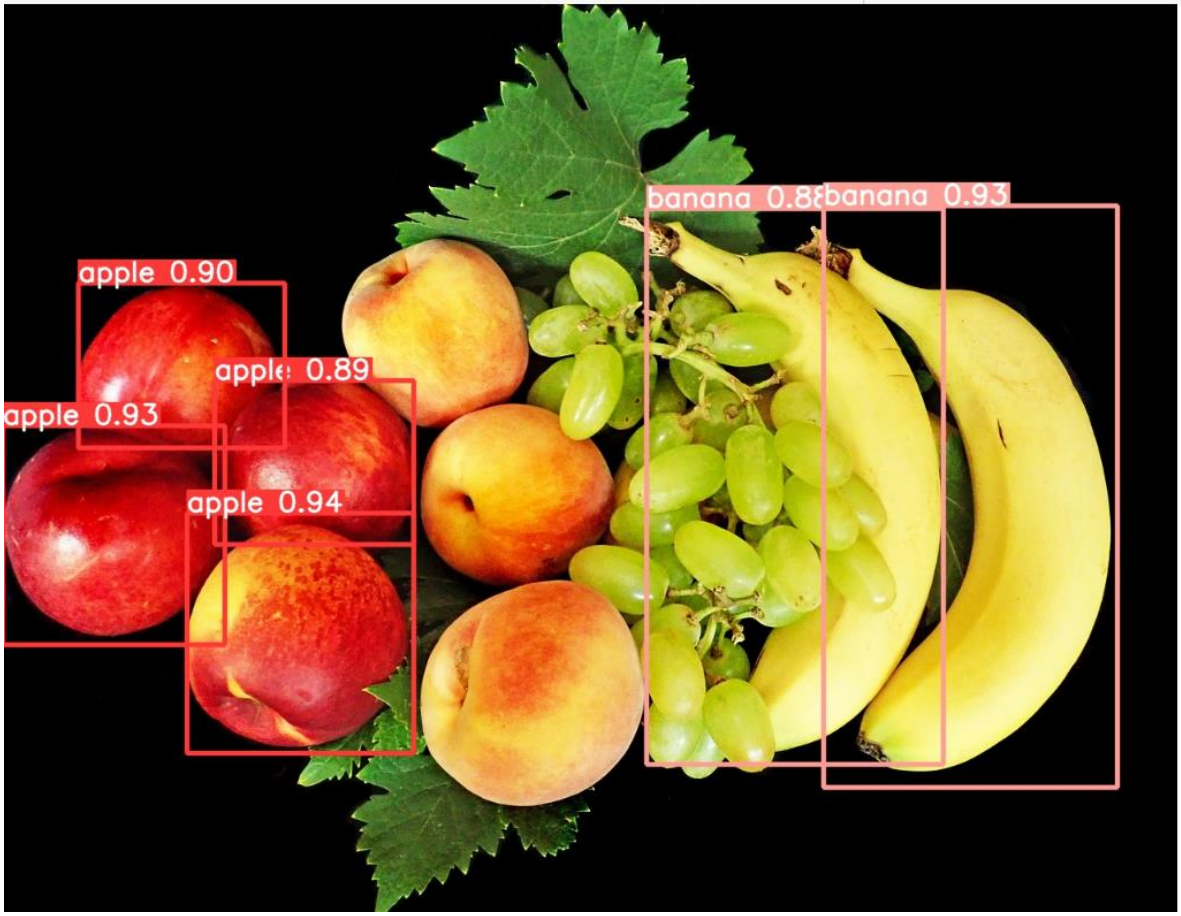


[그림 IV-6] tensorboard를 이용하여 세부 내용 확인

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

```
!python detect.py --weights runs/train/exp/weights/best.pt --img 416 --conf 0.1 --source /content/yolov5/images
```

```
#display inference on ALL test images

import glob
from IPython.display import Image, display

for imageName in glob.glob('/content/yolov5/runs/detect/exp/*.jpg'): #이미지 파일 형식에 맞춰 .png 또는 .jpg 등으로 수정
    display(Image(filename=imageName))
    print("₩n")
```



[그림 IV-7] 학습된 모델 결과 테스트

5. onnx 파싱

- 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 컴파일 시 사용됩니다.

[그림 IV-8] yolov5 모델을 onnx로 내보내기

- Yolov5 학습은 완료되었고 이후에 DFC 가이드를 참고하여 hef 파일로 컴파일하면 됩니다.