

YOLO

ai-contents

Exported on 06/18/2023

Table of Contents

1	개요	3
2	Google Colaboratory	4
3	YOLO v8.....	5
4	실습	6
4.1	Colab열기	6
4.2	런타임(Runtime)유형 변경	7
4.3	google drive 마운트.....	9
4.4	GPU연결 확인	10
4.5	Yolo 설치.....	10
4.6	Dataset 준비	11
4.7	yaml 파일 수정(가장 중요).....	12
4.8	모델 학습	15
4.9	모델로 predict해보기	16
4.10	local에서 모델 활용.....	16

1 개요

- YOLO란, You Only Look Once의 약자로 Object detection 분야에서 많이 알려진 모델입니다.
 - 처음으로 one-stage-detection방법을 고안해 실시간으로 Object Detection이 가능하게 만들었습니다.
 - object detection을 하는 방법론은 많지만 yolo의 가장 큰 장점은 실시간에 적용가능할 정도로 빠르다는 점에 있습니다.
 - 해당 과정에서는 roboflow의 public dataset을 이용해 google colaboratory + YOLO v8을 통하여 진행합니다.
-

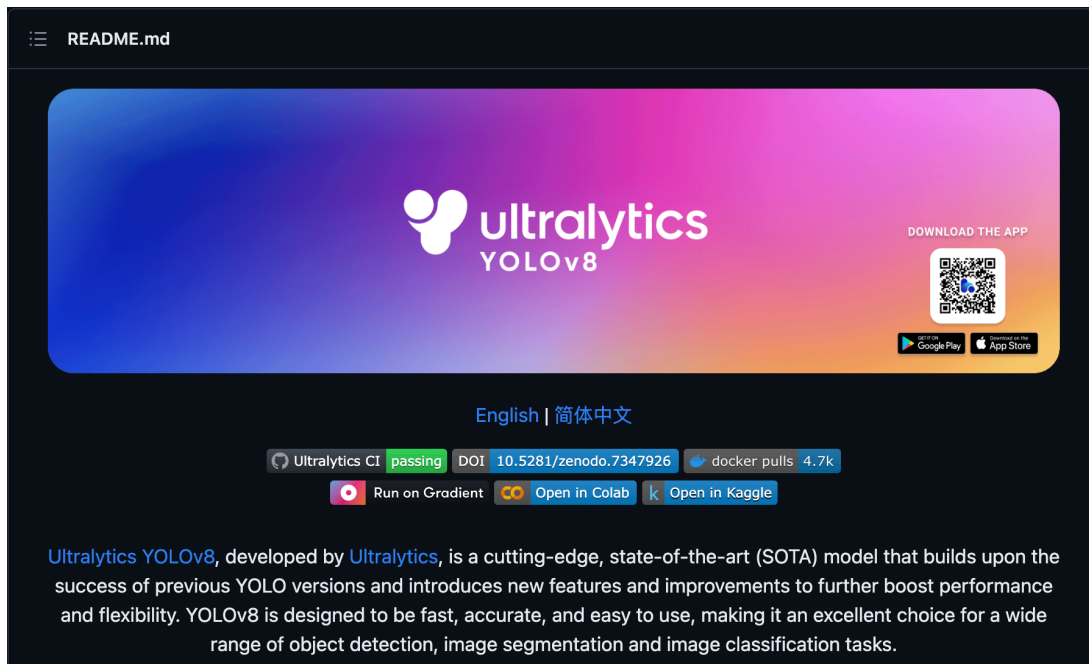
2 Google Colaboratory

- 구글 Colaboratory(이하 Colab)는 클라우드 기반 개발환경입니다.
- 구글은 GPU, TPU, RAM을 사용하여 Jupyter notebook과 같은 환경을 클라우드 서비스로 제공합니다.
- YOLO의 경우, gpu의 성능과 그에 맞는 CUDA 및 cuDNN 설치와 설정 등 각자 local computer에 따라 설정해야 하는 어려움이 있습니다.
- 그에 따른 변수를 최대한 배제하고자 해당 과정은 Colab에서 진행하려 합니다.



3 YOLO v8

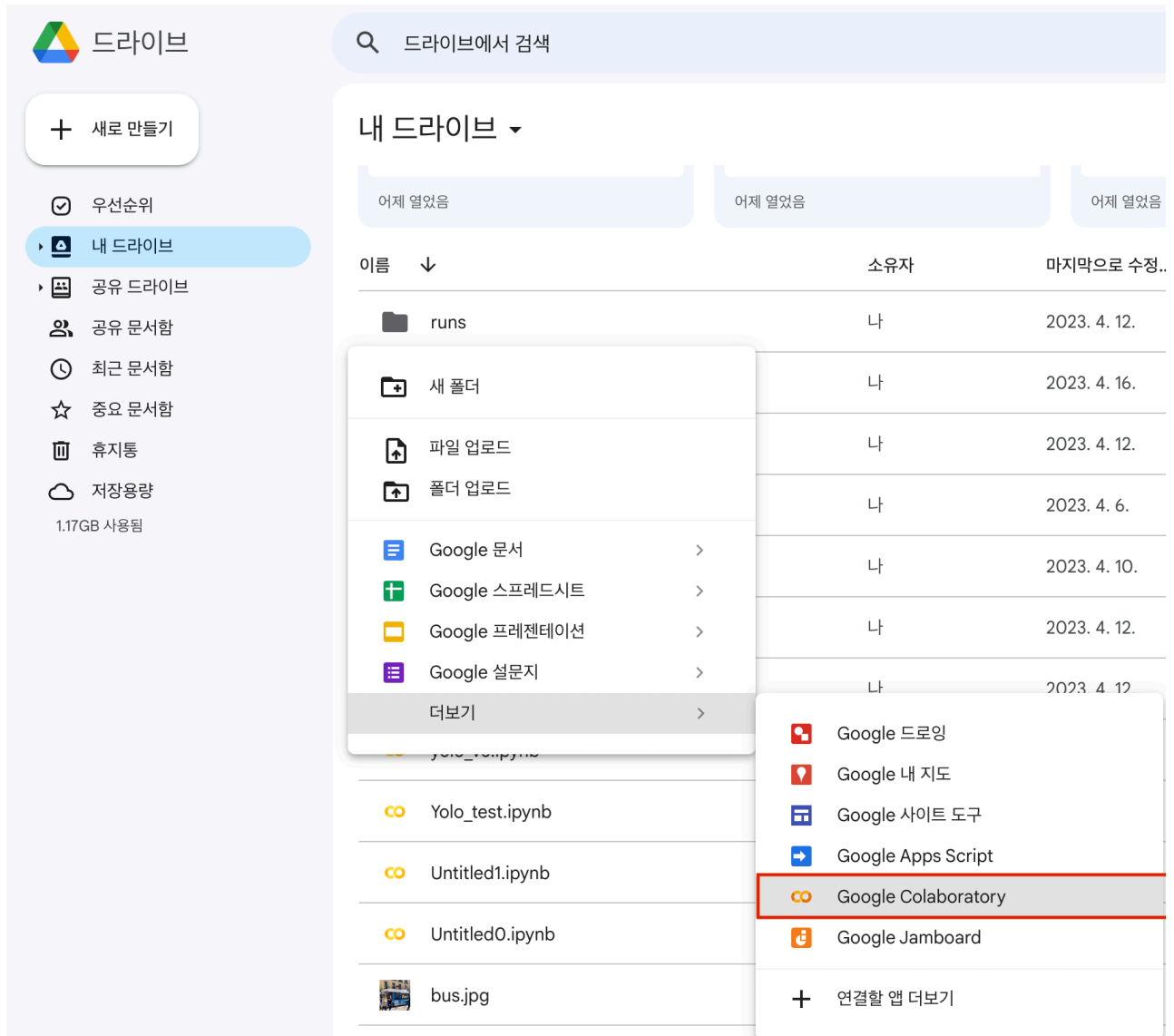
- 앞서 설명했듯이 YOLO는 object detection에서 가장 대중적인 오픈소스 중에 하나입니다.
- 그중 v8의 경우 가장 최신 버전으로, 이를 채택한 이유는 설치과정이나 수정할 사항이 제일 간단합니다.
- 물론 언제나 가장 최신 기술이 좋다고 장담할 수 없지만 여러분들이 대회를 치루는 데에 있어 큰 문제는 없으므로 v8로 진행하고자 합니다.



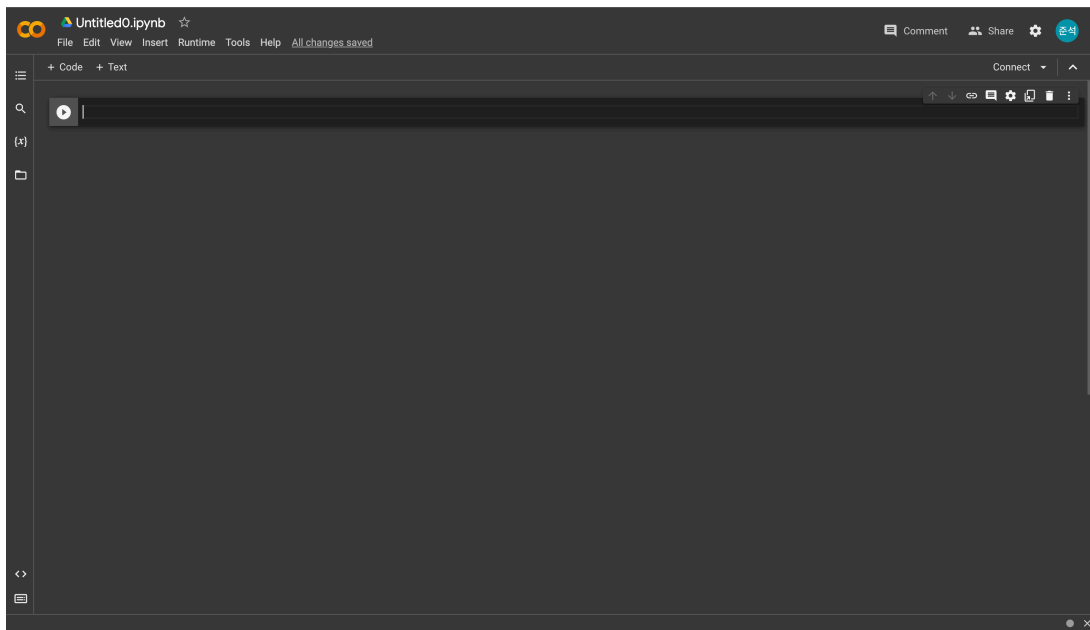
4 실습

4.1 Colab열기

1-1. Google Drive에서 colabotory(빨간색)클릭

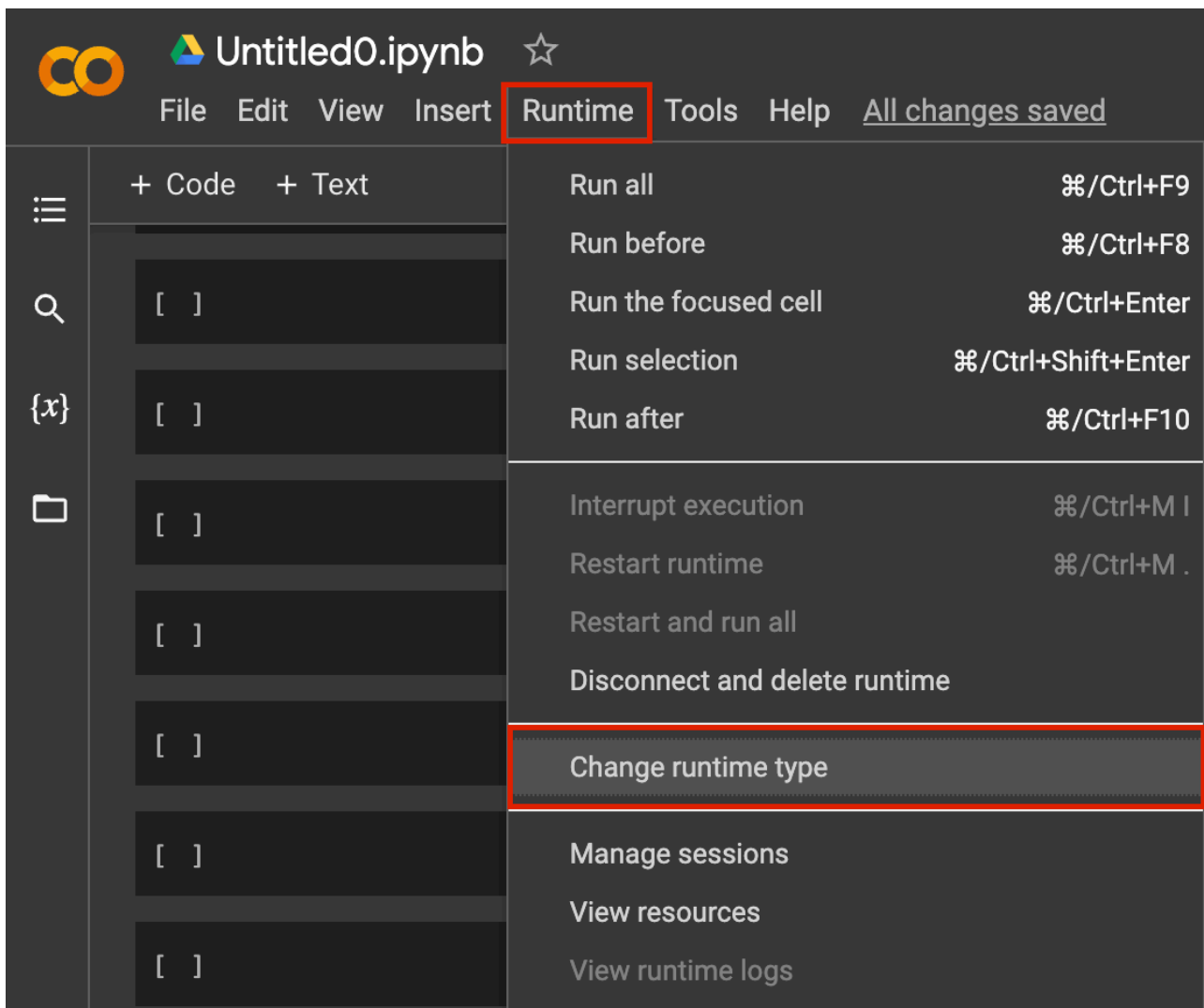


1-2. Colabotory가 열립니다.

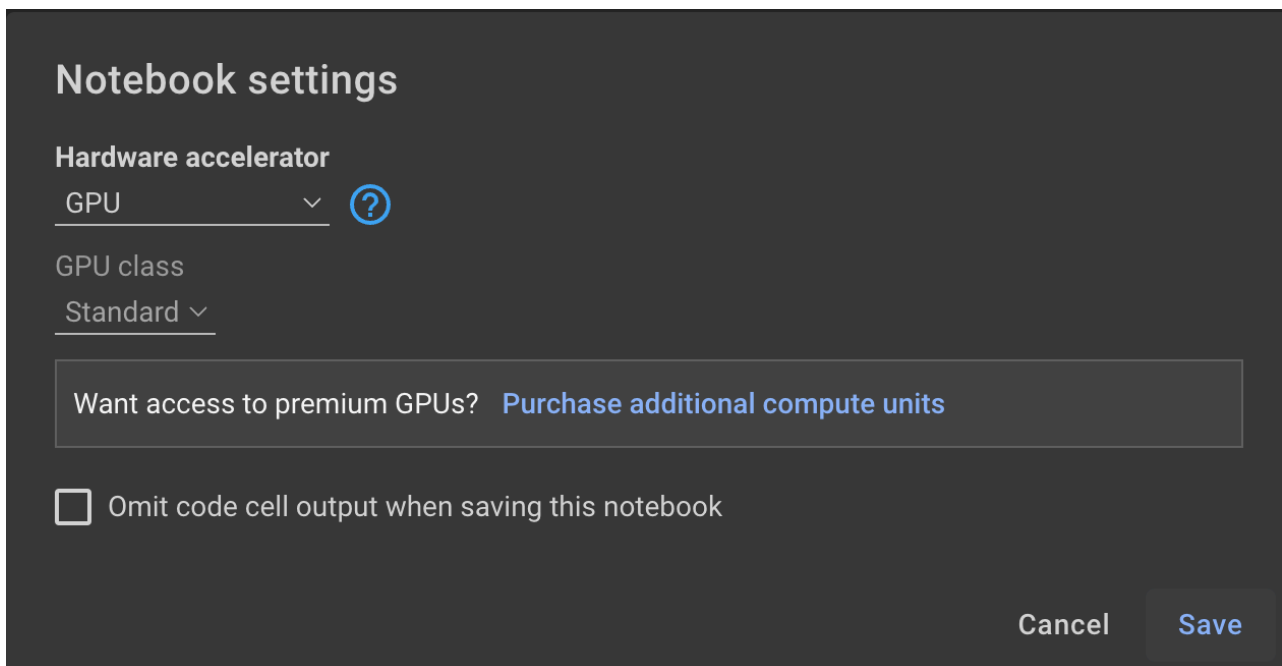


4.2 런타임(Runtime)유형 변경

google에서 지원하는 gpu를 사용하기 위해서 런타임 유형을 변경해 줍니다.

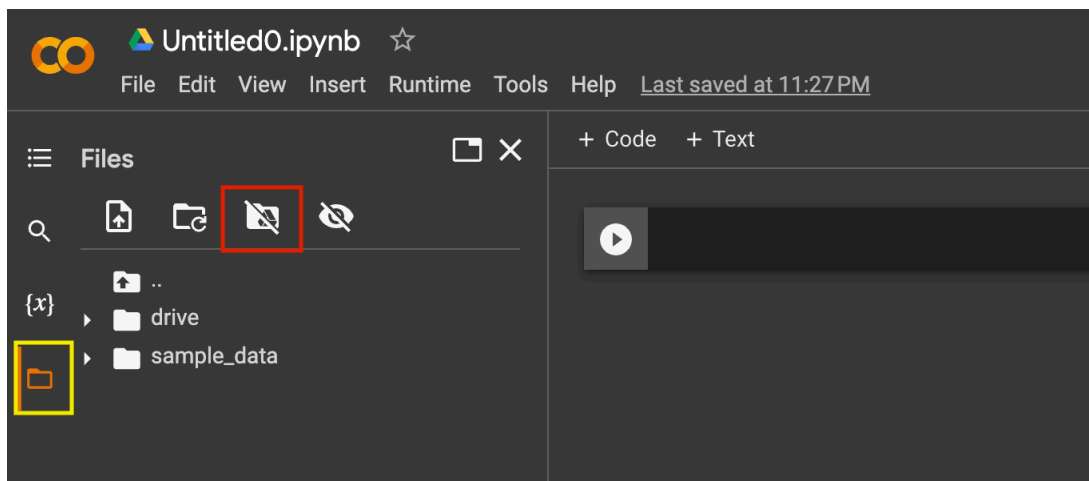


Hardware accelerator를 GPU로 바꿔줍니다.

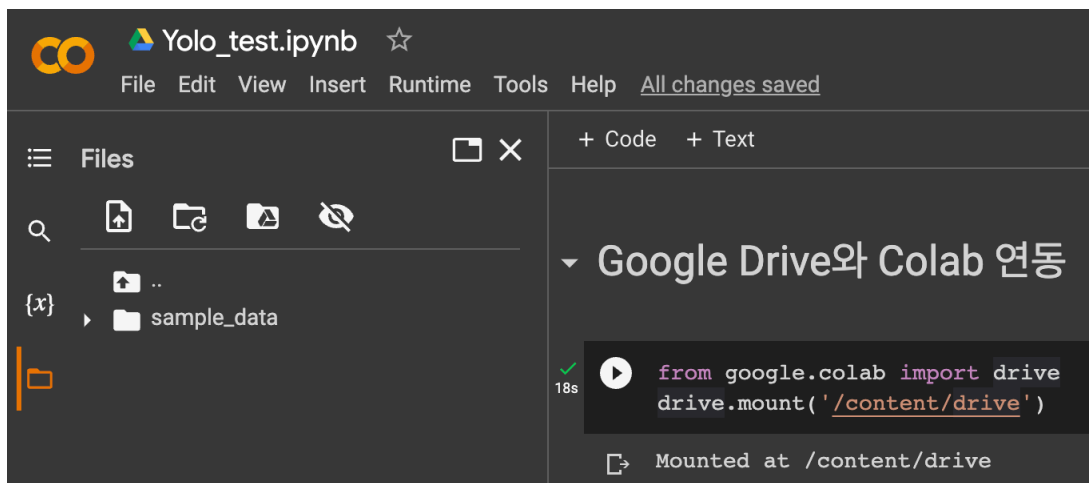


4.3 google drive 마운트

3-1. 왼쪽 메뉴바의 네번째 폴더 그림(노란색)을 누르고 생성되는 메뉴바 세번째 아이콘(빨간색)을 누릅니다.

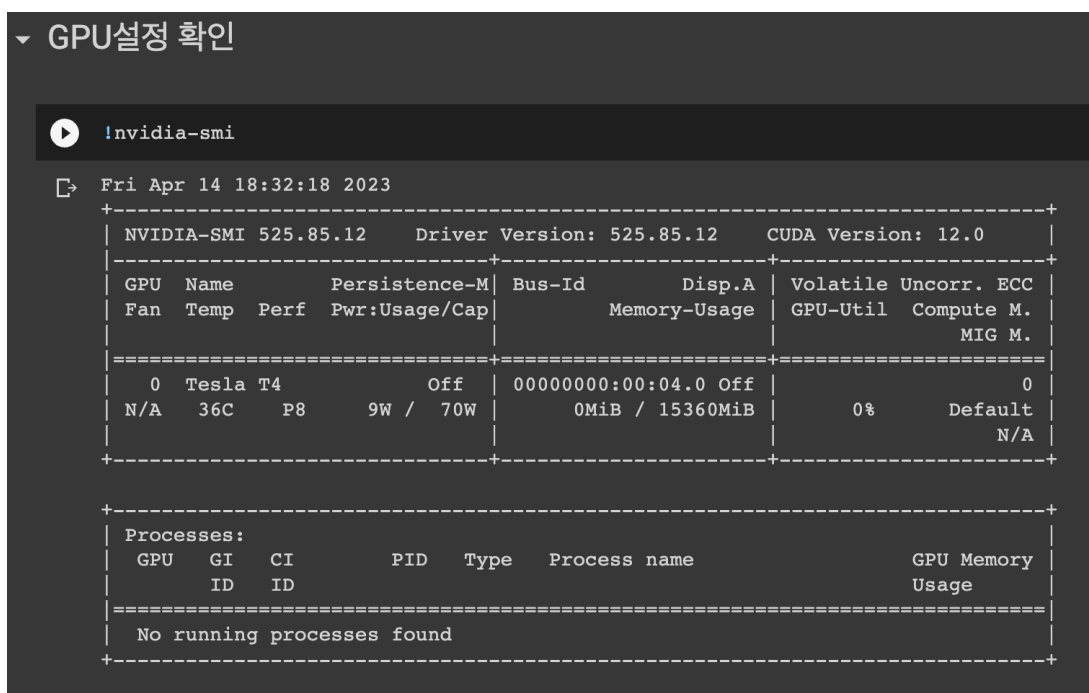


3-2. 아래와 같은 코드가 입력되는 경우도 있습니다.



4.4 GPU연결 확인

!nvidia-smi



4.5 Yolo 설치

!pip install ultralytics

- 해당 과정에서 쓰이는 roboflow dataset에서 요구하는 사양이 ultralytics<= 8.0.20입니다.

```
!pip install ultralytics==8.0.20
```

```
[ ] !pip install ultralytics==8.0.20

Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Collecting ultralytics==8.0.20
  Downloading ultralytics-8.0.20-py3-none-any.whl (261 kB)
    261.2/261.2 kB 10.1 MB/s eta 0:00:00
Requirement already satisfied: torchvision>=0.8.1 in /usr/local/lib/python3.9/dist-packages (from ultralytics==8.0.20) (0.15.1+cu118)
Requirement already satisfied: tqdm>=4.64.0 in /usr/local/lib/python3.9/dist-packages (from ultralytics==8.0.20) (4.65.0)
Requirement already satisfied: seaborn>=0.11.0 in /usr/local/lib/python3.9/dist-packages (from ultralytics==8.0.20) (0.12.2)
Requirement already satisfied: numpy>=1.18.5 in /usr/local/lib/python3.9/dist-packages (from ultralytics==8.0.20) (1.22.4)
Requirement already satisfied: requests>=2.23.0 in /usr/local/lib/python3.9/dist-packages (from ultralytics==8.0.20) (2.27.1)
Requirement already satisfied: scipy>=1.4.1 in /usr/local/lib/python3.9/dist-packages (from ultralytics==8.0.20) (1.10.1)
Requirement already satisfied: pandas>=1.1.4 in /usr/local/lib/python3.9/dist-packages (from ultralytics==8.0.20) (1.5.3)
Requirement already satisfied: thop>=0.1.1 in /usr/local/lib/python3.9/dist-packages (from ultralytics==8.0.20) (0.1.1.post2209072238)
Requirement already satisfied: sentry-sdk in /usr/local/lib/python3.9/dist-packages (from ultralytics==8.0.20) (1.19.1)
Requirement already satisfied: matplotlib>=3.2.2 in /usr/local/lib/python3.9/dist-packages (from ultralytics==8.0.20) (3.7.1)
Requirement already satisfied: torch>=1.7.0 in /usr/local/lib/python3.9/dist-packages (from ultralytics==8.0.20) (2.0.0+cu118)
Requirement already satisfied: PyYAML>=5.3.1 in /usr/local/lib/python3.9/dist-packages (from ultralytics==8.0.20) (6.0)
Requirement already satisfied: opencv-python>=4.6.0 in /usr/local/lib/python3.9/dist-packages (from ultralytics==8.0.20) (4.7.0.72)
Requirement already satisfied: psutil in /usr/local/lib/python3.9/dist-packages (from ultralytics==8.0.20) (5.9.4)
Requirement already satisfied: Pillow>=7.1.2 in /usr/local/lib/python3.9/dist-packages (from ultralytics==8.0.20) (8.4.0)
Requirement already satisfied: tensorboard>=2.4.1 in /usr/local/lib/python3.9/dist-packages (from ultralytics==8.0.20) (2.12.1)
Requirement already satisfied: ipython in /usr/local/lib/python3.9/dist-packages (from ultralytics==8.0.20) (7.34.0)
```

4.6 Dataset 준비

6-1. 이전 강의에서 생성한 download 코드 복사

Your Download Code

Jupyter
 Terminal
 Raw URL

Paste this snippet into [a notebook from our model library](#) » to download and unzip [your dataset](#) »:

```
!pip install roboflow

from roboflow import Roboflow
rf = Roboflow(api_key="")
project = rf.workspace("roboflow-58fyf").project("rock-paper-scissors-sxsw")
dataset = project.version(11).download("yolov8")
```

Warning: Do not share this snippet beyond your team, it contains a private key that is tied to your Roboflow account. Acceptable use policy applies.

Done
Choose a Model

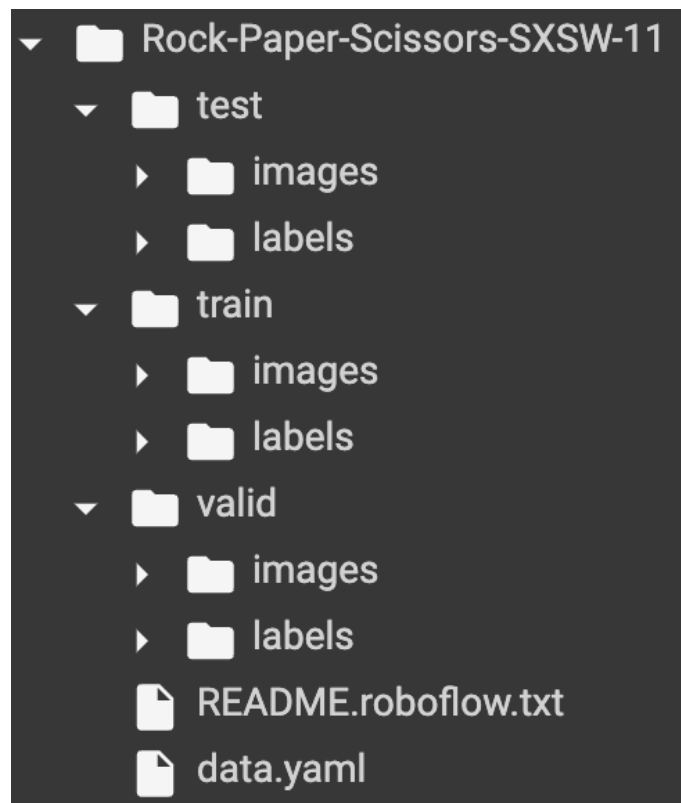
6-2. colab에 붙여넣기 후 실행

```
[3] !pip install roboflow

from roboflow import Roboflow
rf = Roboflow(api_key="qVpgXhb7QbAFkNY6HQdZ")
project = rf.workspace("roboflow-58fyf").project("rock-paper-scissors-sxsw")
dataset = project.version(11).download("yolov8")

Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Collecting roboflow
  Downloading roboflow-1.0.5-py3-none-any.whl (56 kB)
    56.2/56.2 KB 5.6 MB/s eta 0:00:00
Requirement already satisfied: Pillow>=7.1.2 in /usr/local/lib/python3.9/dist-packages (from roboflow) (8.4.0)
Requirement already satisfied: opencv-python>=4.1.2 in /usr/local/lib/python3.9/dist-packages (from roboflow) (4.7.0.72)
Collecting requests-toolbelt
  Downloading requests_toolbelt-0.10.1-py2.py3-none-any.whl (54 kB)
    54.5/54.5 KB 6.5 MB/s eta 0:00:00
Requirement already satisfied: python-dateutil in /usr/local/lib/python3.9/dist-packages (from roboflow) (2.8.2)
Requirement already satisfied: tqdm>=4.41.0 in /usr/local/lib/python3.9/dist-packages (from roboflow) (4.65.0)
Requirement already satisfied: requests in /usr/local/lib/python3.9/dist-packages (from roboflow) (2.27.1)
Requirement already satisfied: urllib3>=1.26.6 in /usr/local/lib/python3.9/dist-packages (from roboflow) (1.26.15)
Requirement already satisfied: chardet==4.0.0 in /usr/local/lib/python3.9/dist-packages (from roboflow) (4.0.0)
Requirement already satisfied: numpy>=1.18.5 in /usr/local/lib/python3.9/dist-packages (from roboflow) (1.22.4)
```

이때 생성된 dataset의 tree구조를 자세히 본다면 custom dataset 또한 어떤식으로 구성해야 할지 알수 있습니다.



1 train, valid는 학습을 위한 폴더, **test**는 모델 성능 검증

4.7 yaml 파일 수정(가장 중요)

- yolo모델이 학습을 하기 위한 데이터들의 경로와 정보를 입력하는 파일

- yolo 학습 시 발생하는 오류 중에 상당수는 yaml파일이 원인인 경우가 많습니다.
- custom data를 만들 때는 직접 yaml파일을 만들어야 합니다.(다운 받은 dataset의 yaml파일의 구조를 참고하시길 바랍니다.)

```

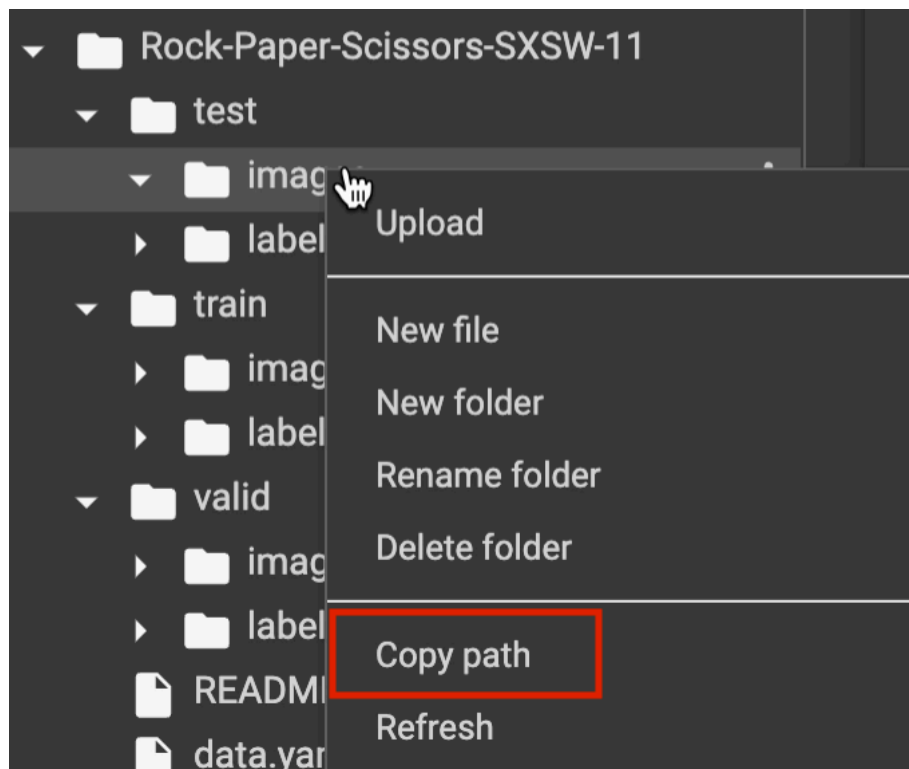
data.yaml ×
1 names:
2 - Paper
3 - Rock
4 - Scissors
5 nc: 3
6 roboflow:
7   license: Private
8   project: rock-paper-scissors-sxsw
9   url: https://app.roboflow.com/roboflow-58fyf/
10  version: 11
11  workspace: roboflow-58fyf
12 test: ../test/images
13 train: Rock-Paper-Scissors-SXSW-11/train/images
14 val: Rock-Paper-Scissors-SXSW-11/valid/images

```

custom dataset을 만들 경우 참고사항

- names - label 종류 // nc - label의 총 개수
- test - 모델 검증을 위한 이미지가 저장된 폴더 // train, val - 학습을 위한 이미지가 저장된 폴더

7-1. test, train, val 폴더 경로 복사



7-2. data.yaml 파일에 각각 붙여넣기

```

data.yaml X
1 names:
2 - Paper
3 - Rock
4 - Scissors
5 nc: 3
6 roboflow:
7   license: Private
8   project: rock-paper-scissors-sxsw
9   url: https://app.roboflow.com/roboflow-58fyf/
10  version: 11
11  workspace: roboflow-58fyf
12 test: /content/Rock-Paper-Scissors-SXSW-11/test
13 train: /content/Rock-Paper-Scissors-SXSW-11/train
14 val: /content/Rock-Paper-Scissors-SXSW-11/valid
15

```

4.8 모델 학습

```

from ultralytics import YOLO

model = YOLO() #pre-trained model을 쓰고 싶으면 model = YOLO('yolov8n.pt')

model.train(data='{data path}/data.yaml', epochs=100) #data.yaml파일이 위치하는 경로

```

epochs란 전체 dataset을 학습한 횟수를 의미합니다.

- 문제집 한권을 하나의 dataset이라고 가정했을때 그 문제집을 처음부터 끝까지 한번 본 경우, epochs=1이라고 볼수 있습니다.
- 이때, epochs를 너무 낮게 설정하면 학습이 제대로 되지 않고 너무 높게 설정하면 시간이 오래 걸리거나 colab의 gpu사용량 제한(약 12시간)에 걸릴 수 있으니 주의하시길 바랍니다.

▶ YOLO모델 학습시키기

```
[ ] from ultralytics import YOLO
```

```
model = YOLO()
```

```
model.train(data='/content/Rock-Paper-Scissors-SXSW-11/data.yaml', epochs=20)
```

	Scissors	588	118	0.757	0.746	0.818	0.485
Epoch	GPU_mem	box_loss	cls_loss	df1_loss	Instances	Size	
8/20	2.87G	1.432	1.734	1.866	6	640: 100%	
	Class	Images	Instances	Box(P	R	mAP50	mAP50-95): 100%
	all	588	406	0.825	0.77	0.841	0.475
	Paper	588	141	0.813	0.677	0.784	0.412
	Rock	588	147	0.776	0.83	0.853	0.48
	Scissors	588	118	0.888	0.804	0.886	0.532
Epoch	GPU_mem	box_loss	cls_loss	df1_loss	Instances	Size	
9/20	2.87G	1.393	1.632	1.818	4	640: 100%	
	Class	Images	Instances	Box(P	R	mAP50	mAP50-95): 100%
	all	588	406	0.822	0.787	0.843	0.481
	Paper	588	141	0.853	0.695	0.81	0.456
	Rock	588	147	0.742	0.859	0.824	0.452
	Scissors	588	118	0.872	0.806	0.895	0.536
Epoch	GPU_mem	box_loss	cls_loss	df1_loss	Instances	Size	
10/20	2.87G	1.363	1.56	1.783	13	640: 100%	
	Class	Images	Instances	Box(P	R	mAP50	mAP50-95): 100%
	all	588	406	0.89	0.783	0.876	0.522
	Paper	588	141	0.921	0.809	0.897	0.517
	Rock	588	147	0.85	0.796	0.858	0.504
	Scissors	588	118	0.9	0.746	0.873	0.547

4.9 모델로 predict해보기

```
!yolo task=detect mode=predict model=/{{model_path}}/best.pt source={{test_image}}
```

```
import locale
locale.getpreferredencoding = lambda: "UTF-8"
```

```
!yolo task=detect mode=predict model=/content/runs/detect/train5/weights/best.pt conf=0.25 source=/content/Rock-Paper-Scissors-SXSW-11/test/images/IMG_7043_MOV-13.jpg
```

```
2023-04-14 21:08:52.754939: W tensorflow/compiler/tf2tensorrt/utils/py_utils.cc:38] TF-TRT Warning: Could not find TensorRT
Ultralytics YOLOv8.0.20 Python-3.9.16 torch-2.0.0+cu118 CUDA:0 (Tesla T4, 15102MiB)
YOLOv8n summary (fused): 168 layers, 3006233 parameters, 0 gradients, 8.1 GFLOPs
image 1/1 /content/Rock-Paper-Scissors-SXSW-11/test/images/IMG_7043_MOV-13.jpg.rf.357623d0c7642153a6890a9650ce4e3e.jpg: 640x640 1 Scissors 7.3ms
Speed: 0.6ms pre-process, 7.3ms inference, 100.6ms postprocess per image at shape (1, 3, 640, 640)
```

학습한 모델로 predict했을 때 아래에 결과가 나타나는 걸(빨간색) 볼 수 있습니다.

4.10 local에서 모델 활용

모델 다운로드

- yolo학습을 끝내면 runs/detect/weights/ 폴더 안에 **best.pt** 와 **last.pt** 이 두가지 모델이 생성됩니다.

- 대개 [best.pt](#) 모델을 다운 받아 사용합니다.

Anaconda에서 명령어

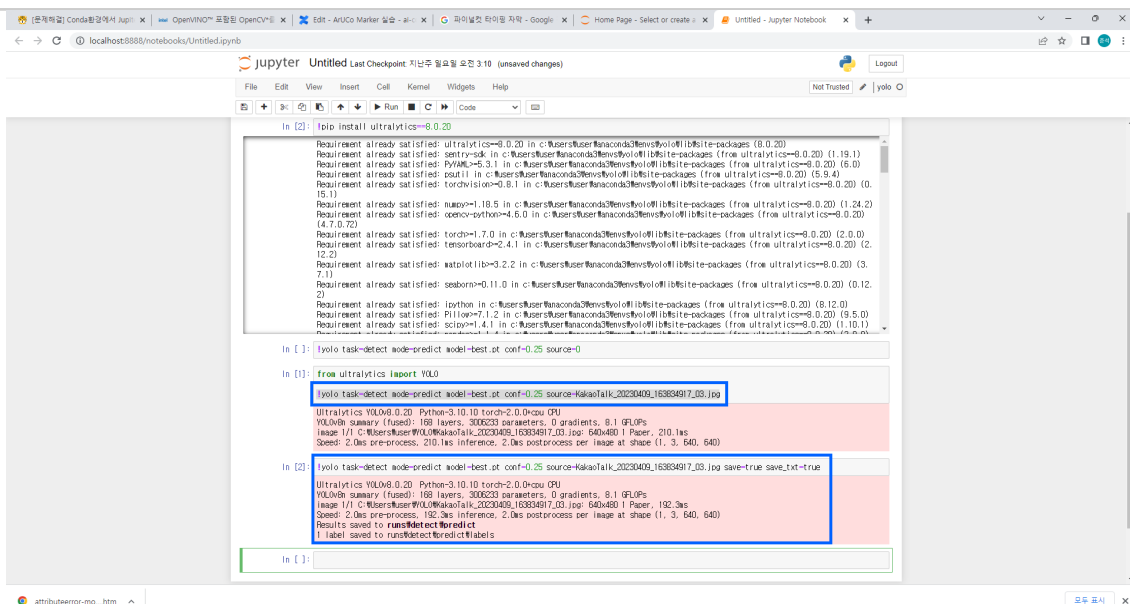
```
cd workspace
jupyter notebook
```

Jupyter notebook 내에서 명령어

```
!pip install ultralytics==8.0.20

#결과를 저장하지 않기
!yolo task=detect mode=predict model={model_path}/best.pt source={test_image}

#결과를 저장하기
!yolo task=detect mode=predict model={model_path}/best.pt source={test_image} \
    save=true save_txt=true
'''
camera를 쓰려면 source=0
save는 모델을 적용한 source파일을 label이 붙은 박스 이미지로 저장
save_txt는 label과 박스의 코너 좌표값이 기록된 텍스트 파일로 저장
'''
```



결과

- 첫번째 박스의 경우, 결과가 3번째 줄에 1 Paper라고 나온 것을 알수 있습니다.
- 두번째 박스의 경우, 위와 동일하지만 결과가 저장된 폴더 위치와 라벨이 저장된 위치까지 보여줍니다.

