

Camera sensor

You Only Look Once (YOLO)

Department of Electrical Engineering, Incheon National University
Hwasu Lee

2024.07.15~17

Contents

1. Overview of YOLO algorithm

- 1) YOLO 알고리즘 개요

2. YOLO v8

- 1) YOLO v8 학습을 위한 Dataset 구축
- 2) YOLO v8 기반 객체 인지 모델 학습
- 3) YOLO v8 기반 객체 인지 모델 검증

3. Github Repository

- 1) Github 구조
- 2) Github Repository 설정

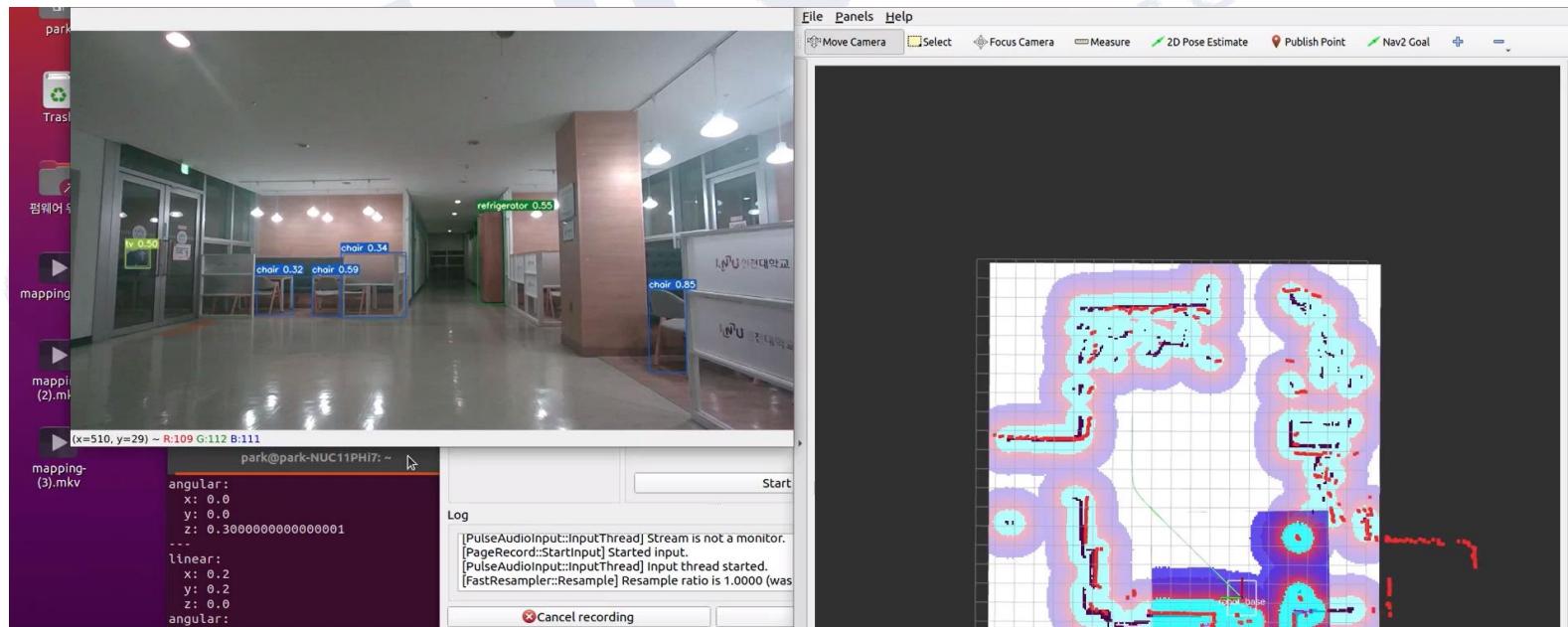
1. Overview of YOLO algorithm

1. Overview of YOLO algorithm

1) YOLO(You Only Look Once) 알고리즘의 개요

- YOLO 알고리즘이란?

- Camera 센서를 통해 Object detection 및 Object classification을 동시에 수행하는 실시간 객체 검출 기법
- 1-stage(Object Classification) 구조를 가지는 YOLO 알고리즘은 2-stage (Region Proposal + Object Classification) 구조를 갖는 R-CNN 계열의 알고리즘 대비 월등한 속도를 가짐
- 이러한 결과로 실시간 Object detection 및 Object Classification이 가능



2. YOLO v8 algorithm 실습

2. YOLO v8 algorithm 실습

1) YOLO v8 학습을 위한 Dataset 구축

- Roboflow Dataset [1 / 7]



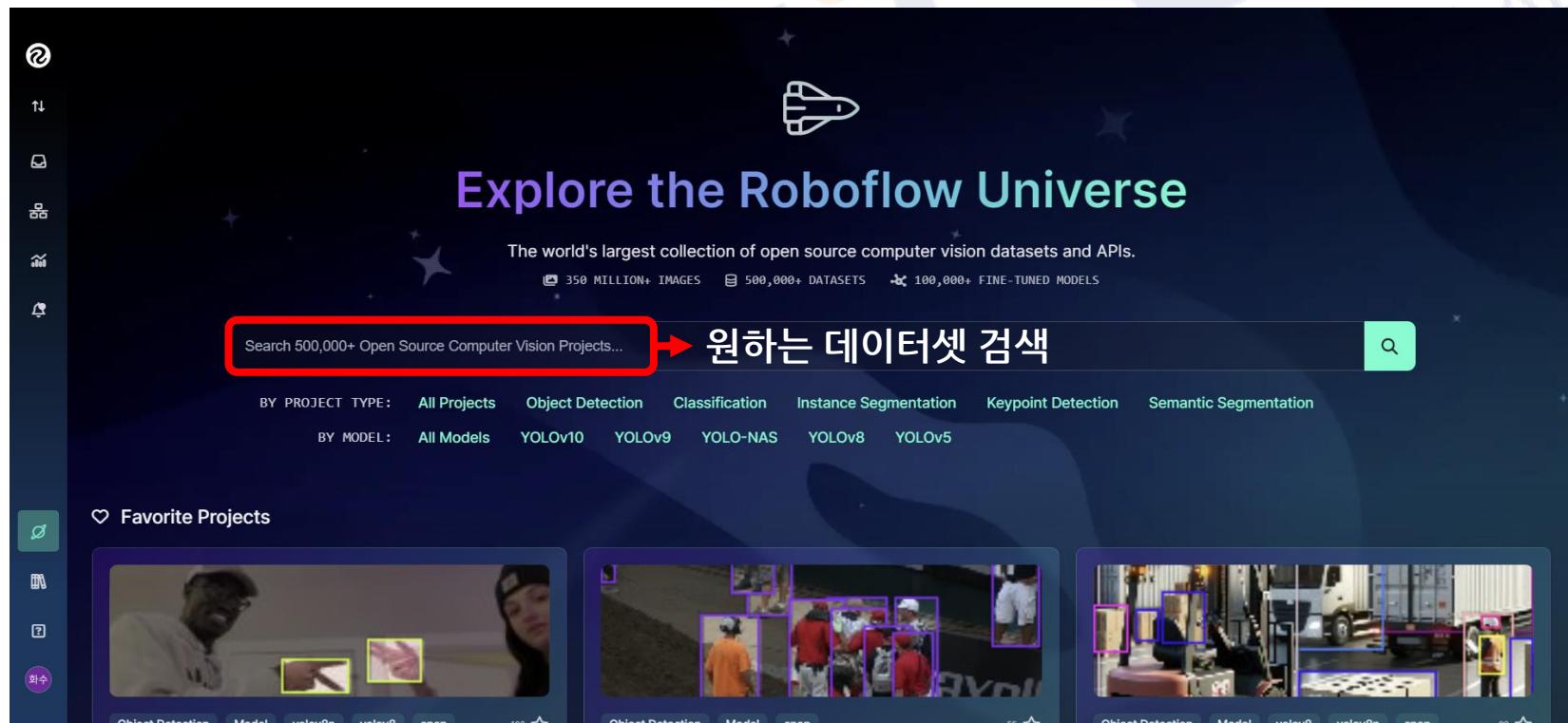
- Computer Vision Application을 구축하고 배포하는데 필요한 툴과 서비스를 제공하는 플랫폼
- Machine Learning과 AI(Artificial Intelligence)를 활용한 이미지 및 영상 데이터 처리에 용이
 - ➔ 데이터셋 관리 : 다양한 이미지 데이터를 손쉽게 업로드, 레이블링, 관리할 수 있는 인터페이스 제공
 - ➔ 데이터 라벨링 : 사용자가 이미지를 라벨링하고 주석을 달 수 있도록 직관적인 툴 제공(객체 감지, 분류 등)
 - ➔ 데이터 증강 : 기존 데이터셋을 다양하게 변형하여 확장할 수 있도록 하는 데이터 증강 툴 제공
 - ➔ 모델 훈련 : 자신의 데이터셋을 기반으로 Machine Learning 모델을 훈련시킬 수 있도록 지원
 - ➔ 모델 배포 : 자신의 데이터셋을 기반으로 훈련된 모델을 쉽게 배포하고 관리할 수 있는 툴 제공
- API를 통해 실시간 추론 기능 또한 구현 가능

2. YOLO v8 algorithm 실습

1) YOLO v8 학습을 위한 Dataset 구축

- Roboflow Dataset [2 / 7]

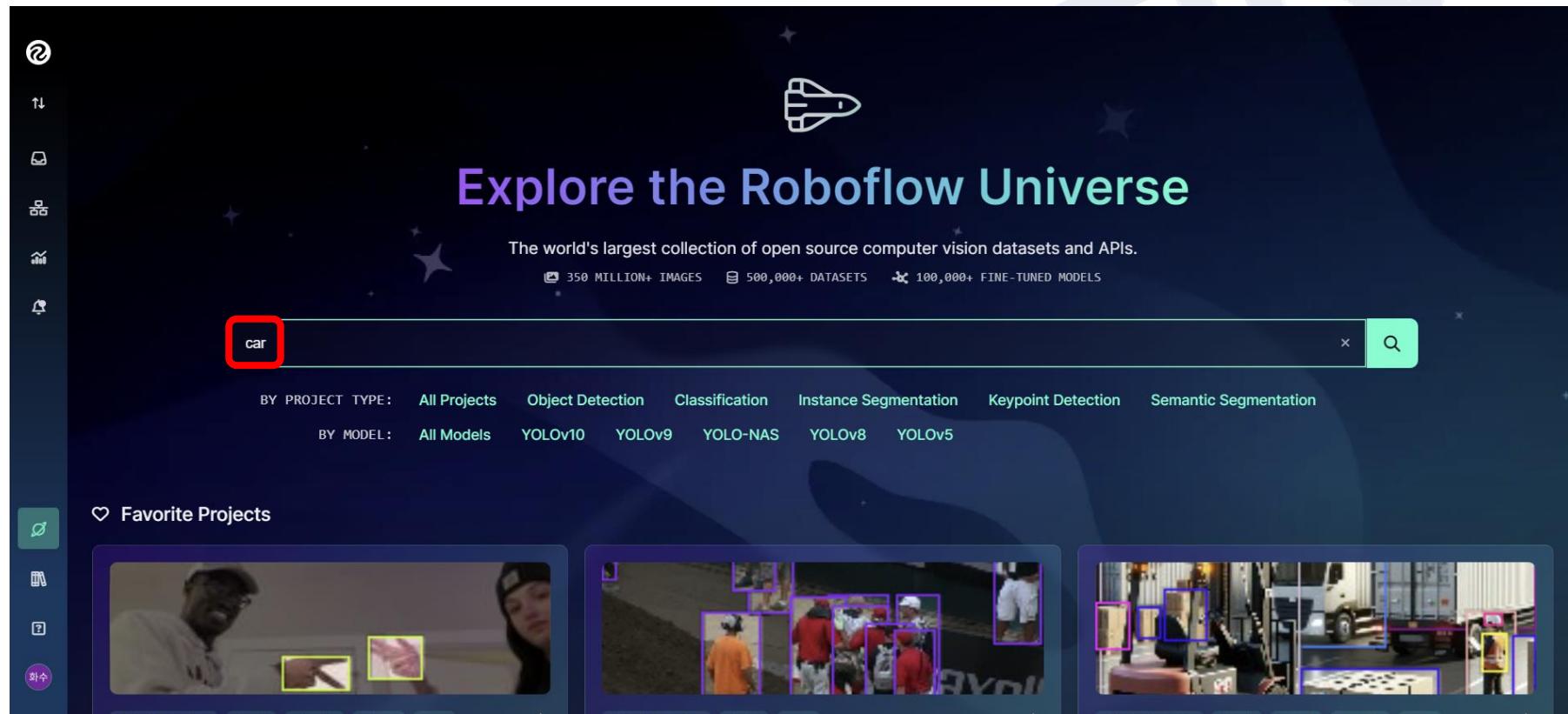
→ <https://universe.roboflow.com/> (해당 링크에 접속하여 원하는 데이터셋 검색)



2. YOLO v8 algorithm 실습

1) YOLO v8 학습을 위한 Dataset 구축

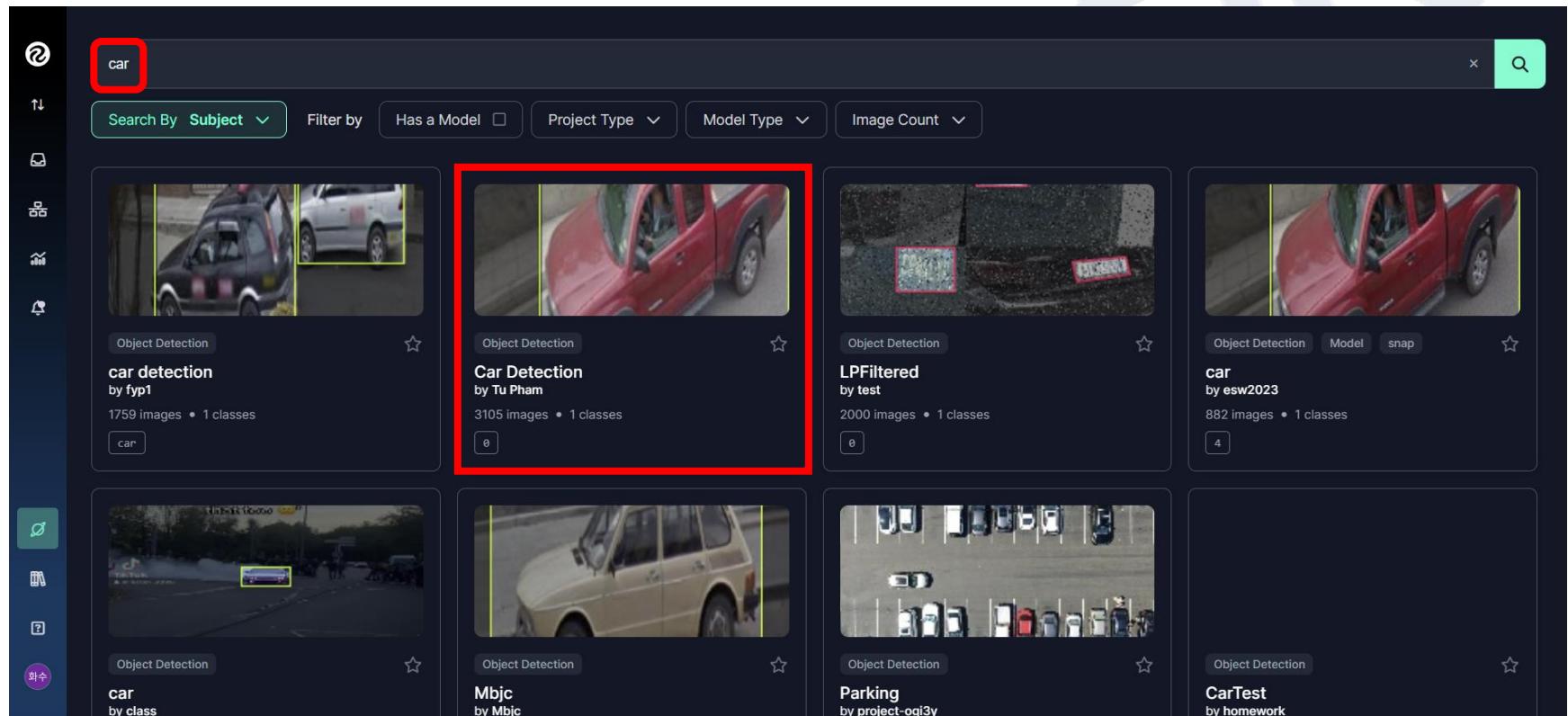
- Roboflow Dataset [3 / 7]



2. YOLO v8 algorithm 실습

1) YOLO v8 학습을 위한 Dataset 구축

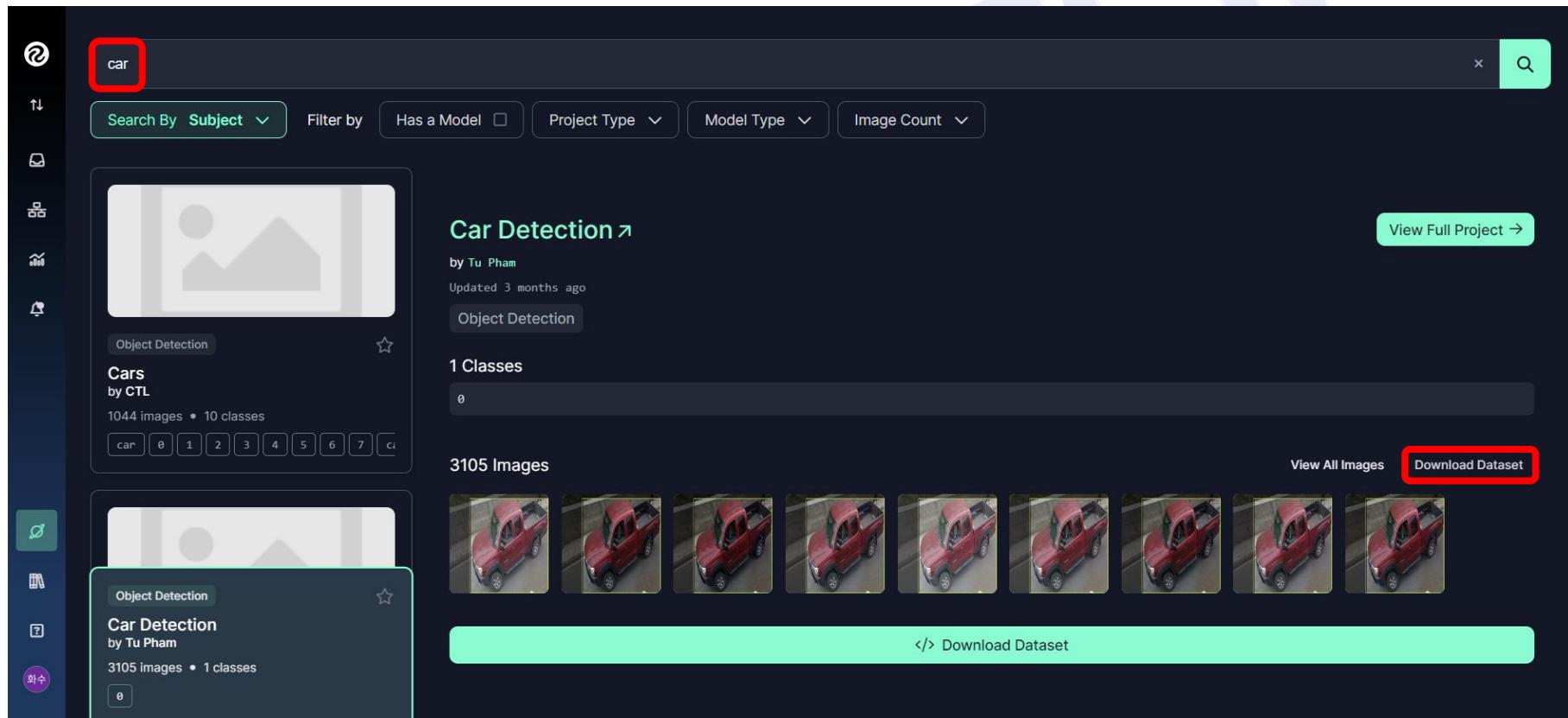
- Roboflow Dataset [4 / 7]



2. YOLO v8 algorithm 실습

1) YOLO v8 학습을 위한 Dataset 구축

- Roboflow Dataset [5 / 7]

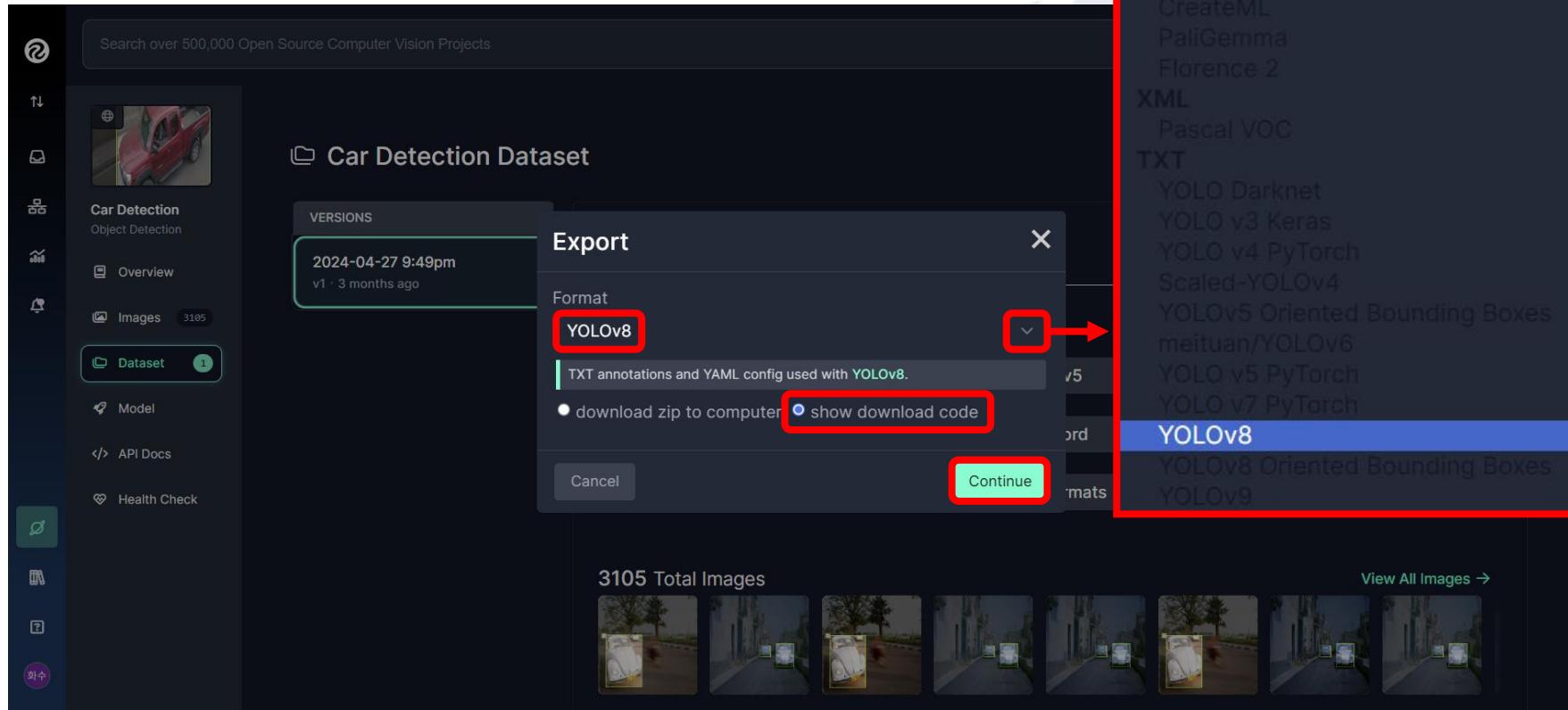


The screenshot shows the Roboflow web interface for searching datasets. The search bar at the top contains the text "car", which is highlighted with a red box. Below the search bar are several filter options: "Search By Subject" (set to "Object Detection"), "Filter by Has a Model" (unchecked), "Project Type" (dropdown), "Model Type" (dropdown), and "Image Count" (dropdown). On the left side, there is a sidebar with various icons and a "화수" (History) button at the bottom. The main content area displays two dataset cards. The first card is for a "Car Detection" project by Tu Pham, updated 3 months ago. It shows 1 Class (Cars by CTL) and 1044 images across 10 classes. The second card is for another "Car Detection" project by Tu Pham, updated 3 months ago, showing 1 Class (Cars by CTL) and 3105 images across 1 class. Both cards feature thumbnail images of red pickup trucks. At the bottom right of the page, there is a large green button labeled "</> Download Dataset".

2. YOLO v8 algorithm 실습

1) YOLO v8 학습을 위한 Dataset 구축

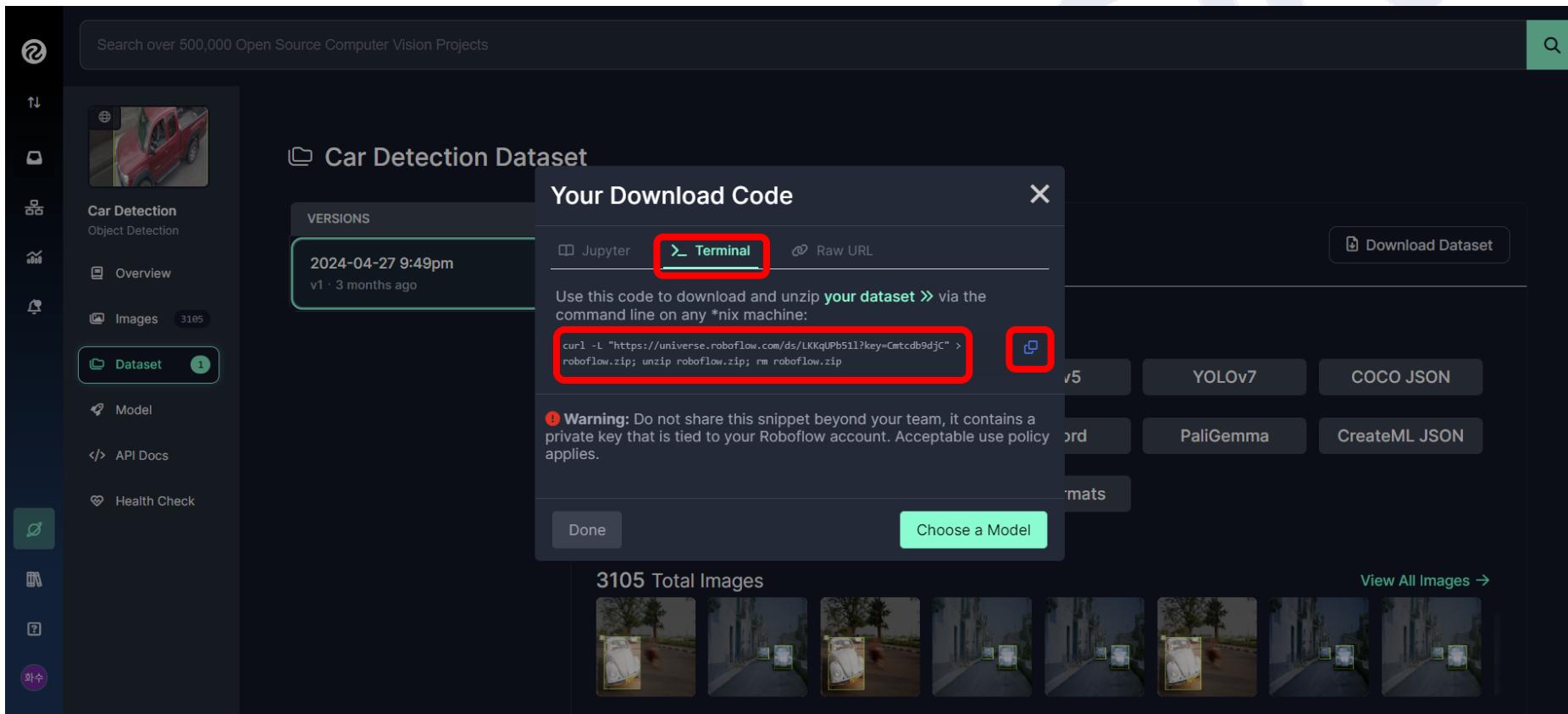
- Roboflow Dataset [6 / 7]



2. YOLO v8 algorithm 실습

1) YOLO v8 학습을 위한 Dataset 구축

- Roboflow Dataset [7 / 7]



The screenshot shows the Roboflow interface for a "Car Detection Dataset". On the left sidebar, there are icons for Home, Projects, Datasets, Models, API Docs, and Health Check. The main area displays the dataset details:

- Dataset Name:** Car Detection
- Type:** Object Detection
- Overview:** 3105 images
- Dataset:** 1 item
- Model:** None
- API Docs:** None
- Health Check:** None

The central part of the screen shows the "Your Download Code" modal. It includes:

- VERSIONS:** 2024-04-27 9:49pm (v1 · 3 months ago)
- Download Options:** Jupyter, Terminal (highlighted with a red box), Raw URL.
- Code Snippet:** curl -L "https://universe.roboflow.com/ds/LKKqUPb511?key=Cmtcdb9djC" > roboflow.zip; unzip roboflow.zip; rm roboflow.zip (highlighted with a red box).
- Copy Button:** A blue copy icon is shown next to the code snippet.
- Warning:** A warning message: "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."
- Buttons:** Done, Choose a Model.
- Image Preview:** A grid of 8 small images showing cars in various environments.

→ 해당 페이지의 Download Code까지 활성화 시켰다면, Dataset 준비가 완료된 것.

2. YOLO v8 algorithm 실습

2) YOLO v8 기반 객체 인지 모델 학습

- Google Colab 환경 구축 [1 / 4]



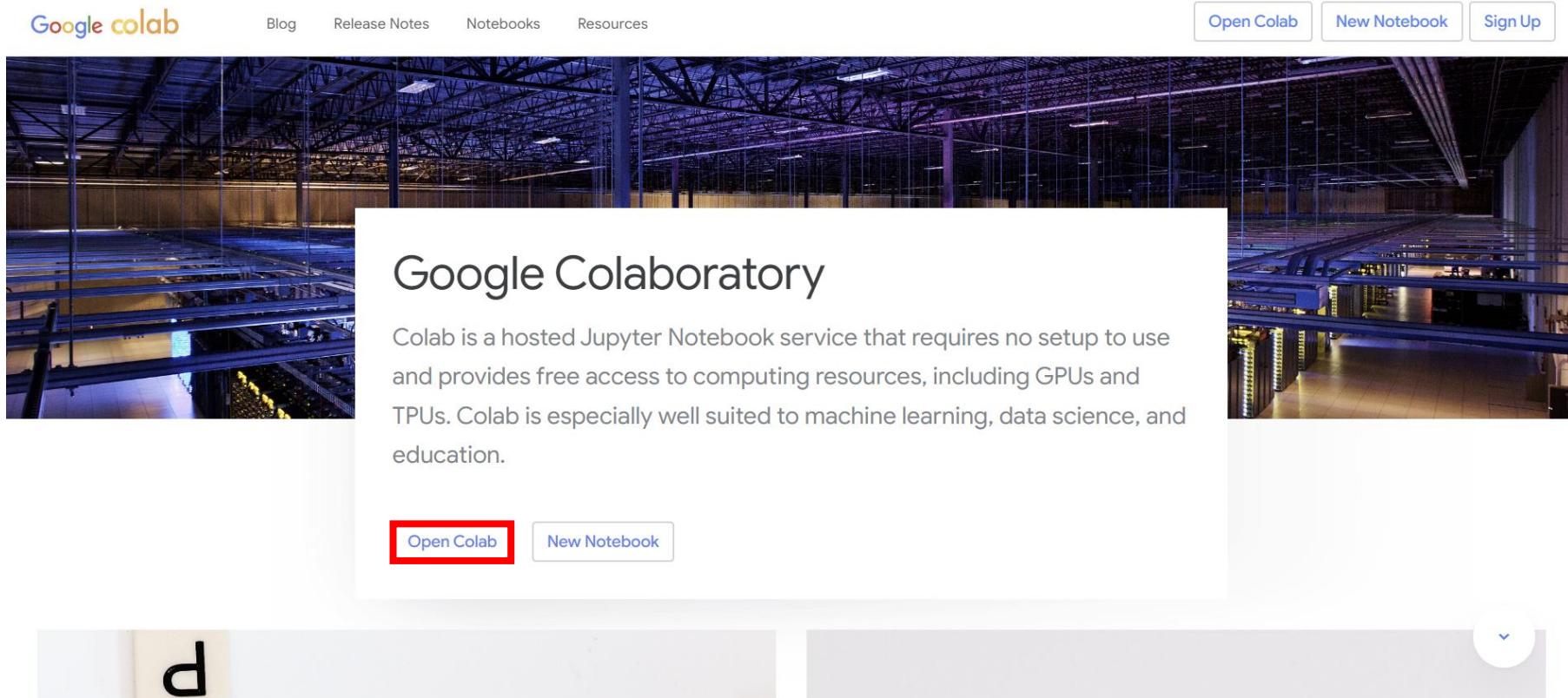
- Google Colab은 Google에서 제공하는 클라우드 기반의 Jupyter 노트북 환경
- Colab을 통해 사용자는 별도의 소프트웨어 설치 없이 웹 브라우저에서 코드를 작성하고 실행 가능
- 가장 큰 특징으로 GPU, TPU를 무료로 사용 가능 (단, 리소스가 제한되어 있음)
- Data Science, Machine Learning, Deep Learning과 같은 프로젝트에 유용

2. YOLO v8 algorithm 실습

2) YOLO v8 기반 객체 인지 모델 학습

- Google Colab 환경 구축 [2 / 4]

→ <https://colab.google/> (해당 링크에 접속하여 Google Colab 실행)

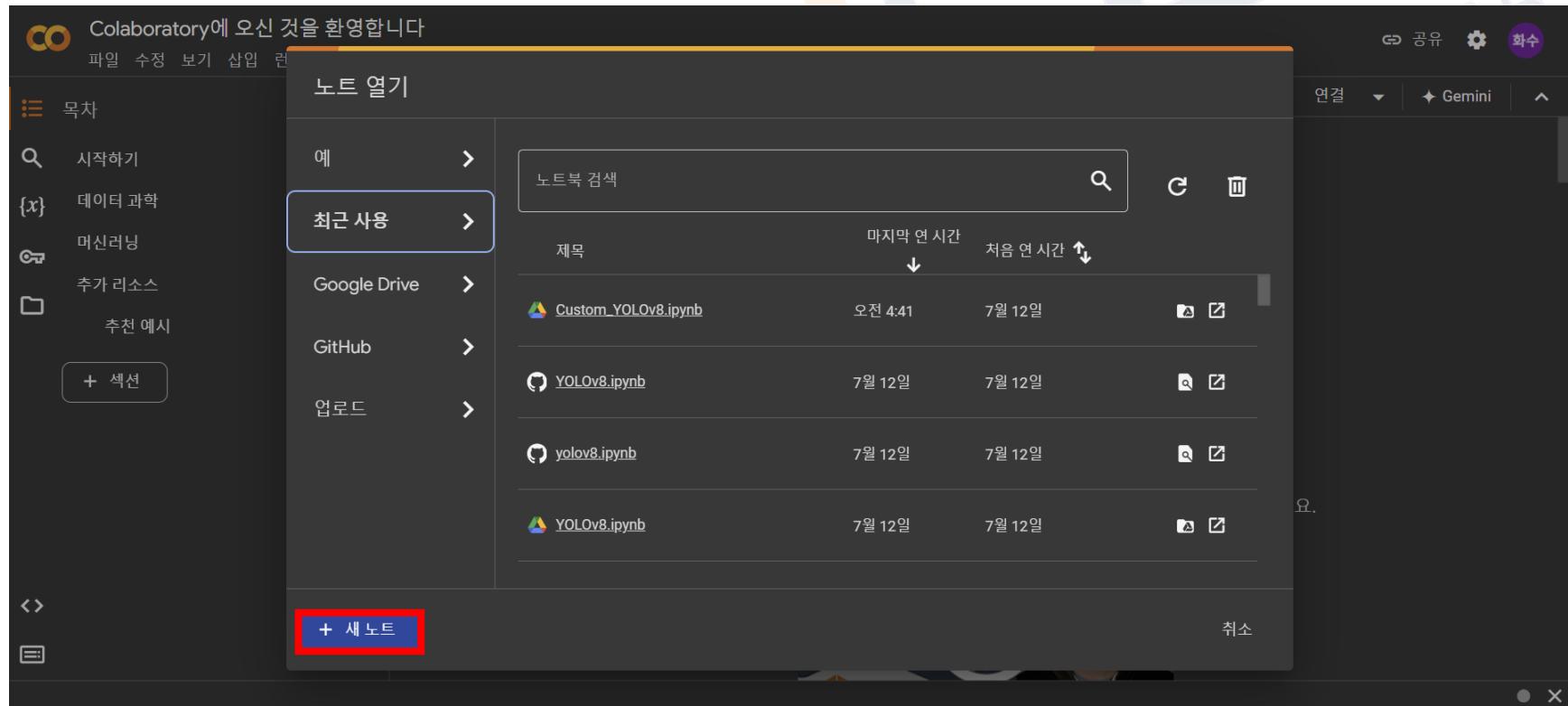


2. YOLO v8 algorithm 실습

2) YOLO v8 기반 객체 인지 모델 학습

- Google Colab 환경 구축 [3 / 4]

→ Google Colab 실행 후 , Google 로그인 및 “새 노트” 클릭하여 Colab 환경 구축

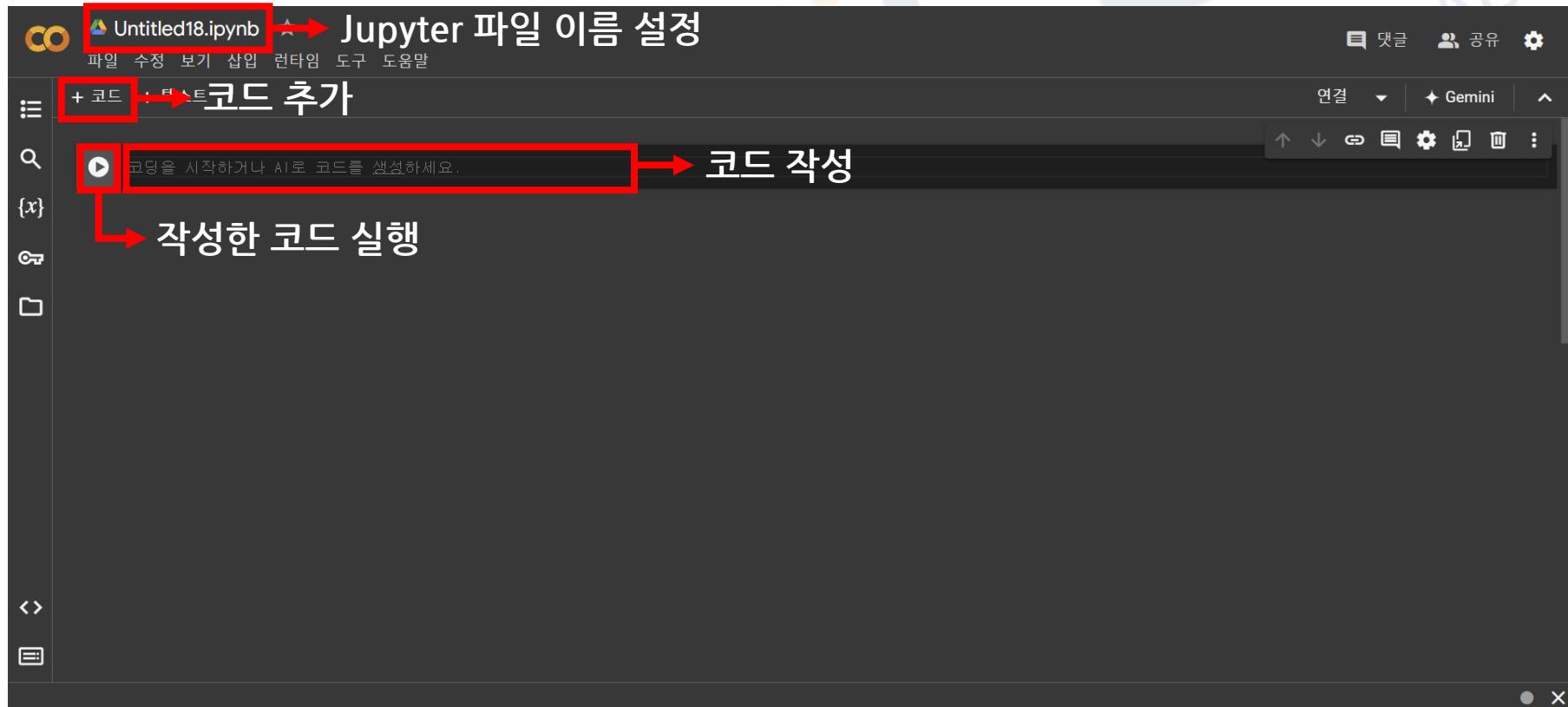


2. YOLO v8 algorithm 실습

2) YOLO v8 기반 객체 인지 모델 학습

- Google Colab 환경 구축 [4 / 4]

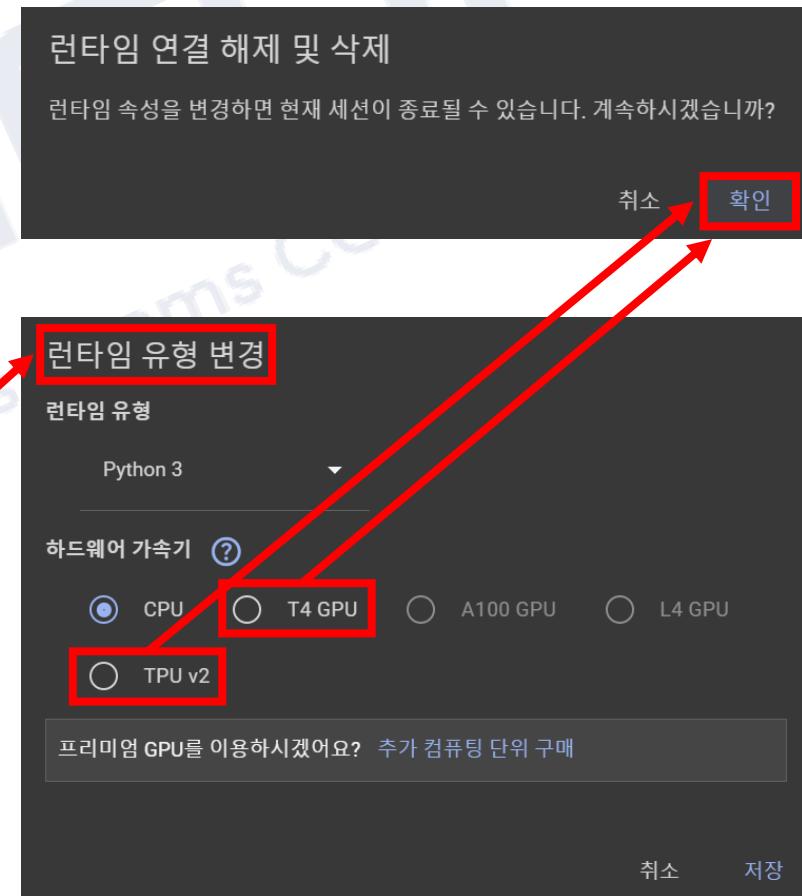
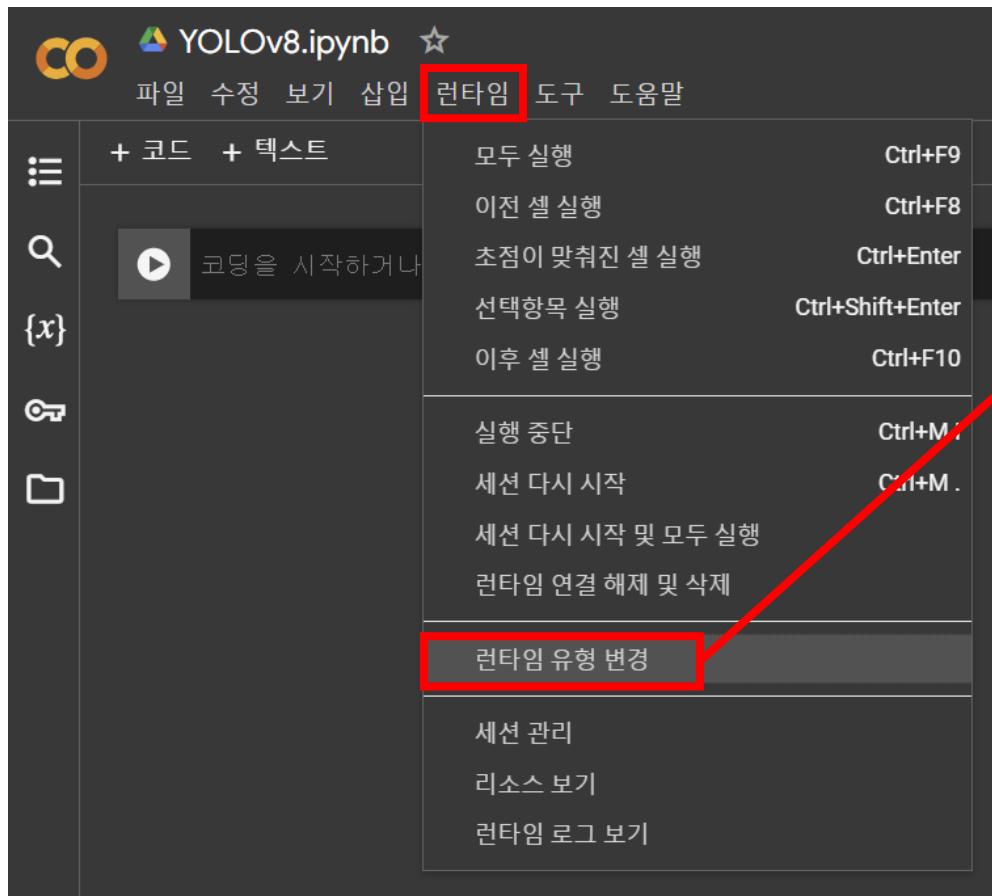
➔ Google Colab 실행 후 , Google 로그인 및 “새 노트” 클릭하여 Colab 환경 구축



2. YOLO v8 algorithm 실습

2) YOLO v8 기반 객체 인지 모델 학습

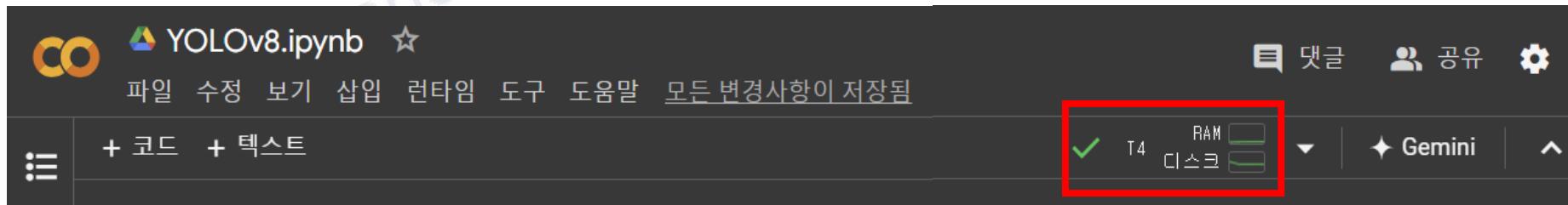
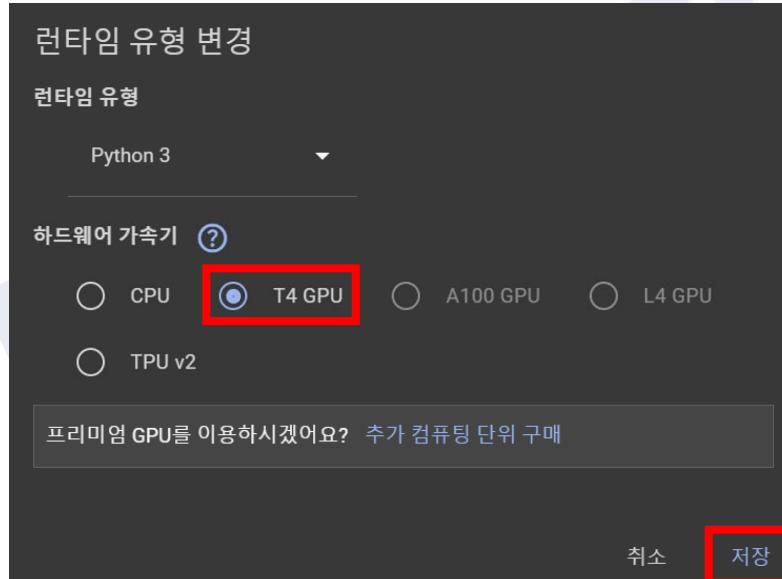
- Google Colab 환경 내 GPU/TPU 설정 [1 / 2]



2. YOLO v8 algorithm 실습

2) YOLO v8 기반 객체 인지 모델 학습

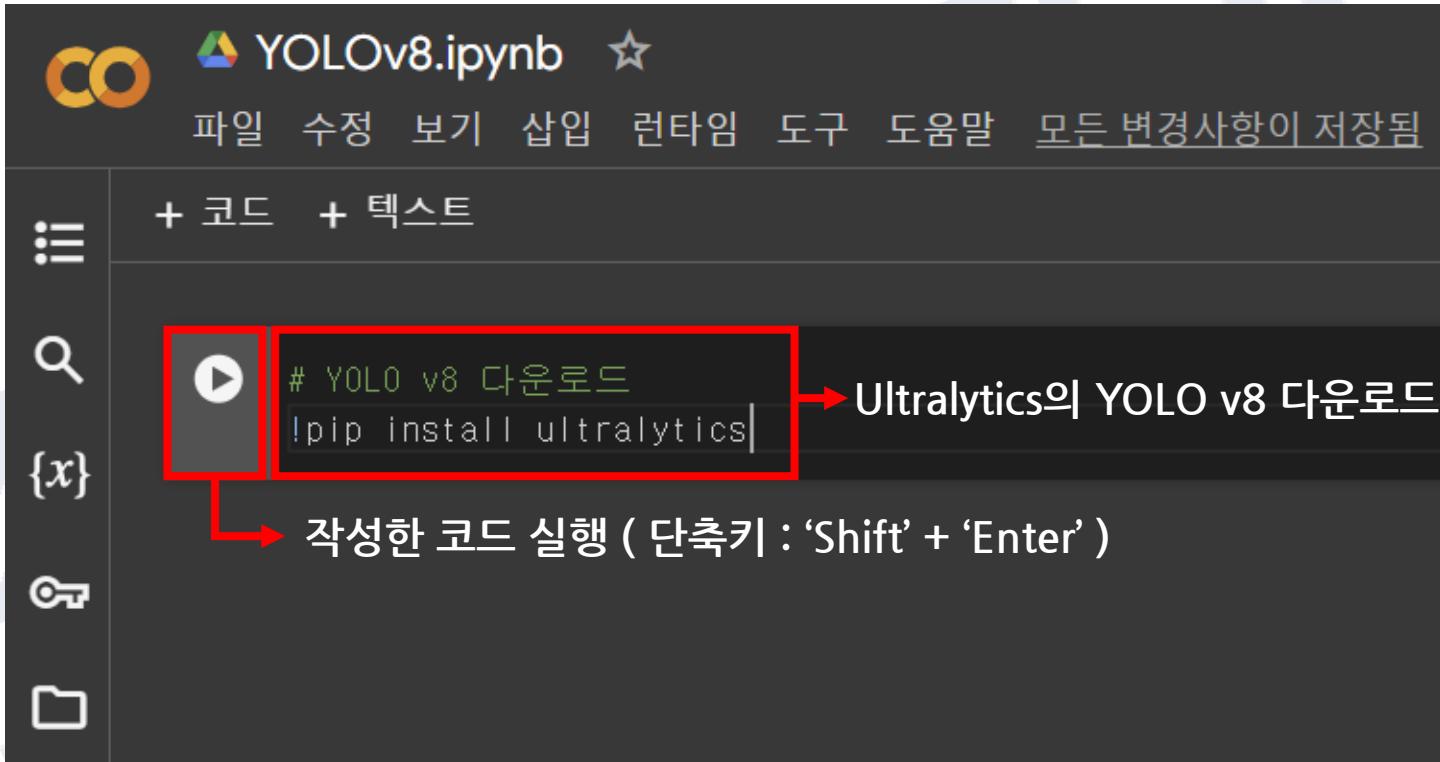
- Google Colab 환경 내 GPU/TPU 설정 [2 / 2]



2. YOLO v8 algorithm 실습

2) YOLO v8 기반 객체 인지 모델 학습

- Google Colab 환경 내 YOLO v8 설치 [1 / 1]



The screenshot shows a Google Colab notebook titled "YOLOv8.ipynb". The code cell contains the following command:

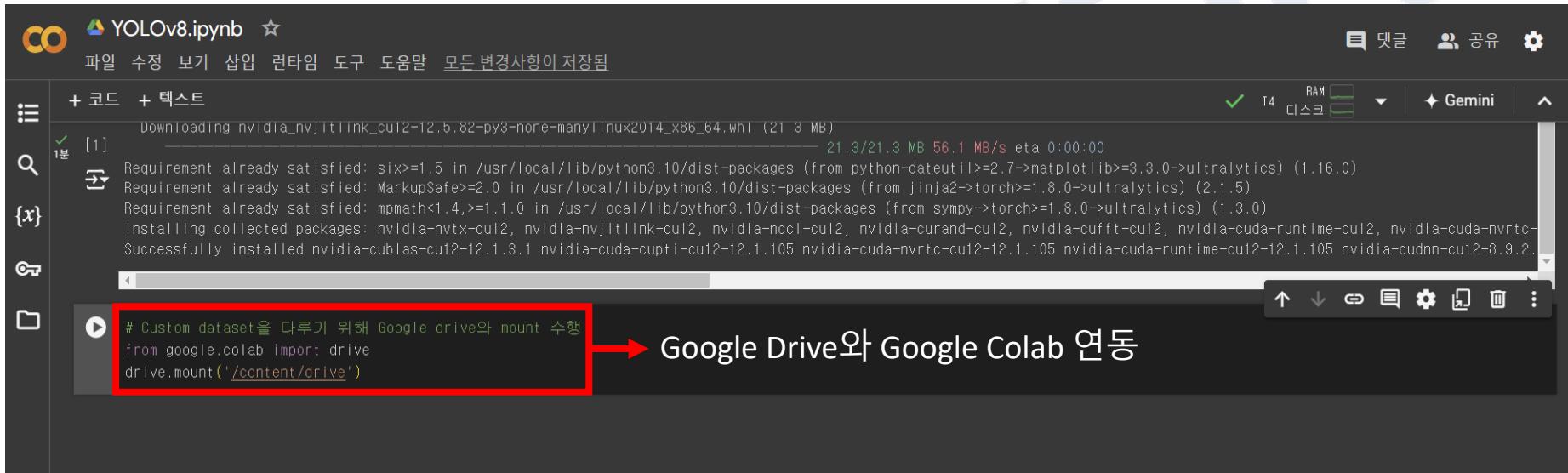
```
# YOLO v8 다운로드  
!pip install ultralytics
```

A red box highlights the play button icon and the code cell. A red arrow points from the right side of the code cell to the text "Ultralytics의 YOLO v8 다운로드". Another red arrow points from the bottom of the code cell to the text "작성한 코드 실행 (단축키 : 'Shift' + 'Enter')".

2. YOLO v8 algorithm 실습

2) YOLO v8 기반 객체 인지 모델 학습

- Google Colab 환경과 Google Drive 연동 [1 / 2]



The screenshot shows a Google Colab notebook titled "YOLOv8.ipynb". The code cell contains the following Python code:

```
# Custom dataset을 다루기 위해 Google drive와 mount 수행
from google.colab import drive
drive.mount('/content/drive')
```

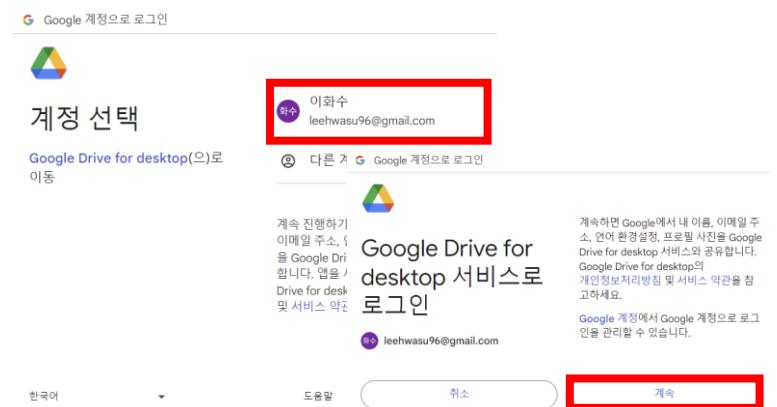
A red box highlights this code. An arrow points from this box to the text "Google Drive와 Google Colab 연동" (Mounting Google Drive and Google Colab).

노트북에서 Google Drive 파일에 액세스하도록 허용하시겠습니까?

이 노트북에서 Google Drive 파일에 대한 액세스를 요청합니다. Google Drive에 대한 액세스 권한을 부여하면 노트북에서 실행되는 코드가 Google Drive의 파일을 수정할 수 있게 됩니다. 이 액세스를 허용하기 전에 노트북 코드를 검토하시기 바랍니다.

아니요

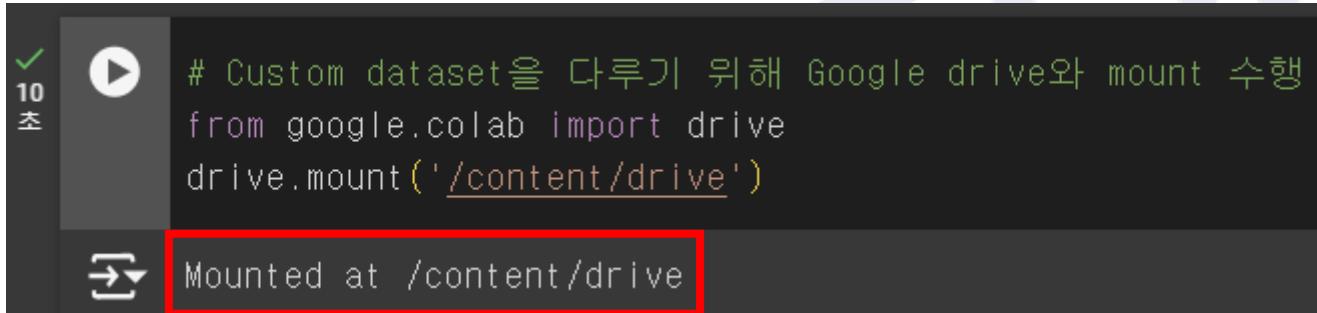
Google Drive에 연결



2. YOLO v8 algorithm 실습

2) YOLO v8 기반 객체 인지 모델 학습

- Google Colab 환경과 Google Drive 연동 [2 / 2]



Custom dataset을 다루기 위해 Google drive와 mount 수행
from google.colab import drive
drive.mount('/content/drive')

Mounted at /content/drive



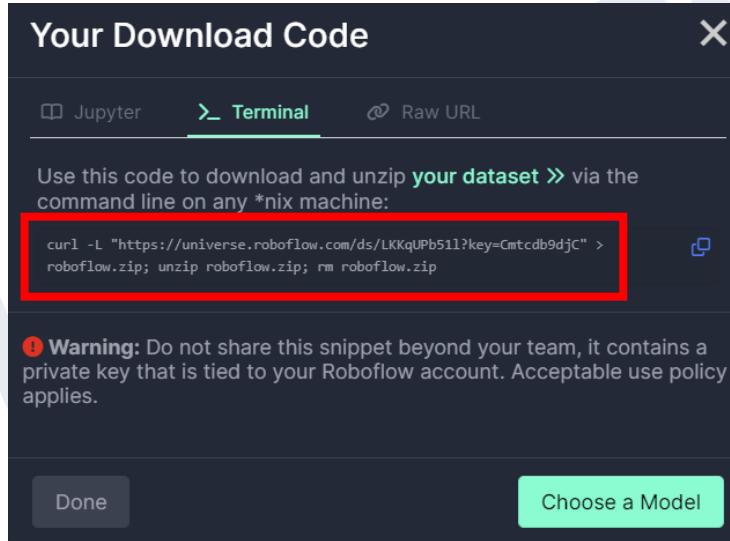
현재 경로 확인
!pwd

/content

2. YOLO v8 algorithm 실습

2) YOLO v8 기반 객체 인지 모델 학습

- Google Colab 환경 내 YOLO v8 Dataset 다운로드 [1 / 4]



Google Drive 내 /content 폴더 안에 Dataset 저장

```
# Roboflow에서 데이터셋 다운로드
# https://universe.roboflow.com/
!curl -L "https://universe.roboflow.com/ds/BtC00zxaeS?key=eQVDTueP22" -o /content/roboflow.zip
```

Roboflow Download Code (맨 앞에 “!” 붙이기)

2. YOLO v8 algorithm 실습

2) YOLO v8 기반 객체 인지 모델 학습

- Google Colab 환경 내 YOLO v8 Dataset 다운로드 [2 / 4]

```
✓ 2초 [11] # Roboflow에서 데이터셋 다운로드  
# https://universe.roboflow.com/  
!curl -L "https://universe.roboflow.com/ds/BtC00zxaeS?key=eQVDTueP22" -o /content/roboflow.zip  
  
→ % Total    % Received % Xferd  Average Speed   Time     Time     Time  Current  
          Dload  Upload   Total   Spent    Left  Speed  
 100  903  100  903    0      0  3293       0  --::-- --::-- --::--  3283  
 100 52.5M  100 52.5M    0      0 24.0M       0  0:00:02  0:00:02  --::-- 40.0M  
  
✓ 0초 [12] # 현재 경로인 /content 폴더 내 자료 확인  
!ls  
  
→ drive roboflow.zip sample_data
```

2. YOLO v8 algorithm 실습

2) YOLO v8 기반 객체 인지 모델 학습

- Google Colab 환경 내 YOLO v8 Dataset 다운로드 [3 / 4]

```
✓ 2초 # 압축 해제
!unzip /content/roboflow.zip -d /content/roboflow

extracting: /content/roboflow/valid/images/-8561227C-3731-4CFC-9FE9-2715B581C159-png_jpg.rf.201e62177e83756dc7d34018f844f31f.jpg
extracting: /content/roboflow/valid/images/-8561227C-3731-4CFC-9FE9-2715B581C159-png_jpg.rf.346d71f70aae413a5c17dd029e5b15d8.jpg
extracting: /content/roboflow/valid/images/-B3FF9CBC-6C9F-4BDD-B814-790723E32E98-png_jpg.rf.60e0e64d7f612613c1e8646fdc4d9e62.jpg
extracting: /content/roboflow/valid/images/-B3FF9CBC-6C9F-4BDD-B814-790723E32E98-png_jpg.rf.cb6d8d1024790c5fc396c660d4364c64.jpg
extracting: /content/roboflow/valid/images/-B41BBCCE-A512-4CBA-94C0-D236EC1DEBA1-png_jpg.rf.a55eb4d80d52cea42844133ec0bc449d.jpg
extracting: /content/roboflow/valid/images/-B83337D2-1F91-4346-B7A4-222F8A45CBF0-png_jpg.rf.48267143d49a33e7230cecebeac85dbf.jpg
extracting: /content/roboflow/valid/images/-B83337D2-1F91-4346-B7A4-222F8A45CBF0-png_jpg.rf.ab272d030df28dad8ed8215df3b1e8d2.jpg
extracting: /content/roboflow/valid/images/-BABD5091-6597-40E1-82D1-29A0F4DCD9E0-png_jpg.rf.a01e27aaa610cd5fa069924ad437b437.jpg
extracting: /content/roboflow/valid/images/-BAE1D45C-2B9A-4C23-AB1D-BD0C25964678-png_jpg.rf.847508c220de18bf5a5e216744fa8e3a.jpg
extracting: /content/roboflow/valid/images/-BAE1D45C-2B9A-4C23-AB1D-BD0C25964678-png_jpg.rf.baf4aae646257ce300023e7f736a8c9a.jpg
extracting: /content/roboflow/valid/images/-BBC6D213-1075-48CE-B772-FDD44F34A3E7-png_jpg.rf.ae7c07a046483eadf6556fc17b90815c.jpg
```

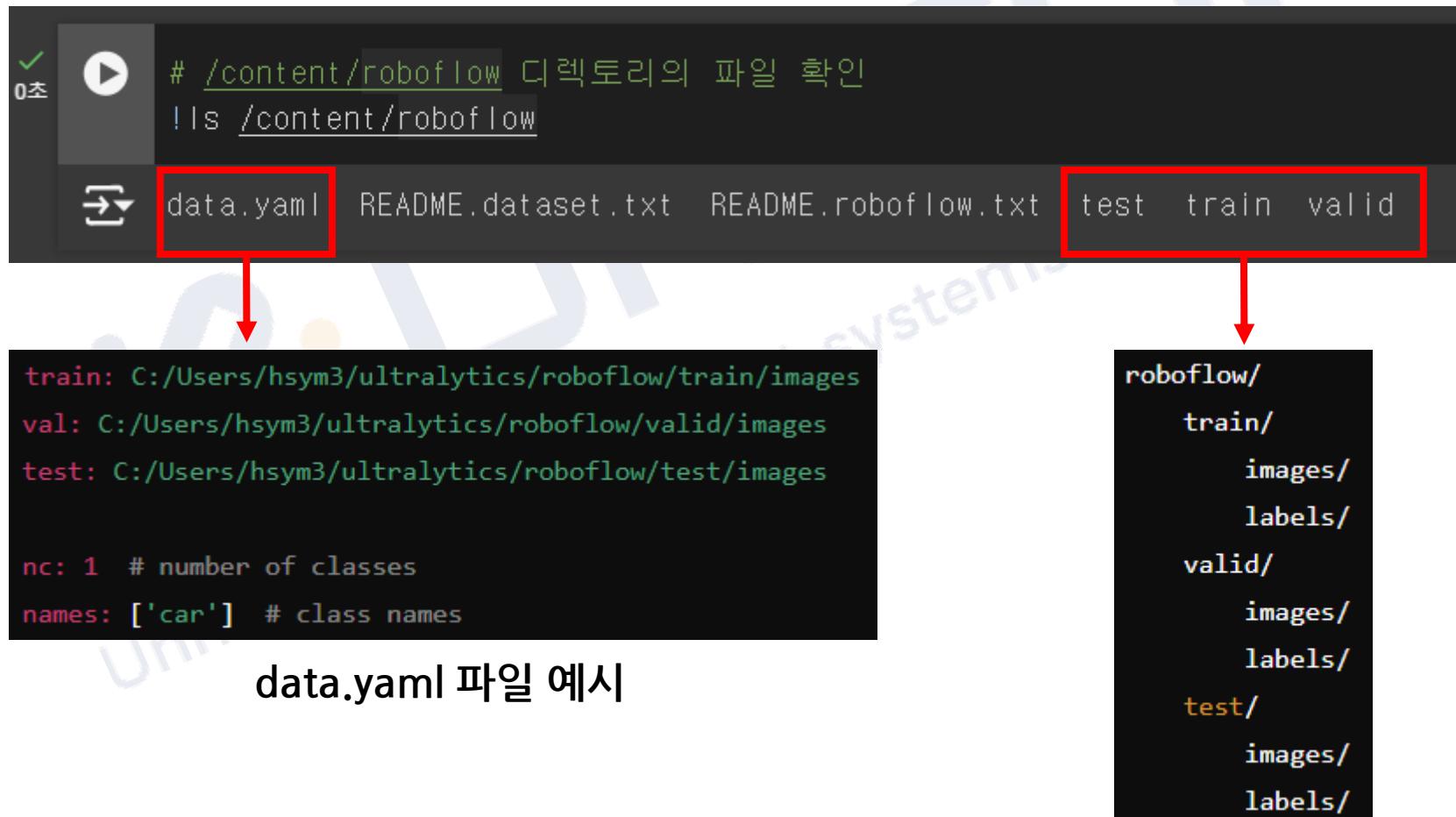
```
[14] ✓ 0초 # 압축 파일 삭제
!rm /content/roboflow.zip

!ls
drive roboflow sample_data
```

2. YOLO v8 algorithm 실습

2) YOLO v8 기반 객체 인지 모델 학습

- Google Colab 환경 내 YOLO v8 Dataset 다운로드 [4 / 4]



```
# /content/roboflow 디렉토리의 파일 확인
!ls /content/roboflow
```

data.yaml README.dataset.txt README.roboflow.txt test train valid

train: C:/Users/hsym3/ultralytics/roboflow/train/images
val: C:/Users/hsym3/ultralytics/roboflow/valid/images
test: C:/Users/hsym3/ultralytics/roboflow/test/images

nc: 1 # number of classes
names: ['car'] # class names

roboflow/
 train/
 images/
 labels/
 valid/
 images/
 labels/
 test/
 images/
 labels/

data.yaml 파일 예시

2. YOLO v8 algorithm 실습

2) YOLO v8 기반 객체 인지 모델 학습

- Google Colab 환경 내 YOLO v8 Dataset 학습 [1 / 6]

```
▶ from ultralytics import YOLO
    import os

    # 필요한 디렉토리 생성
    os.makedirs('/content/runs', exist_ok=True)

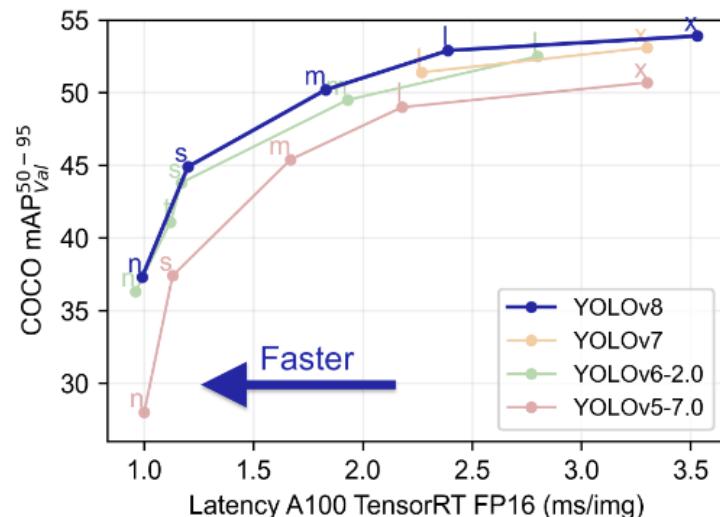
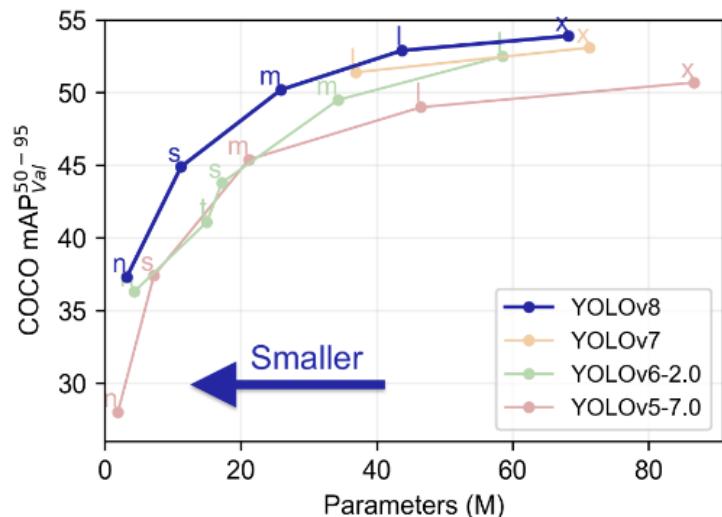
    # YOLOv8 모델 로드
    model = YOLO('yolov8n.yaml')          # 'yolov8n.yaml'은 YOLOv8의 기본 설정 파일

    # 모델 학습
    model.train(
        data='/content/roboflow/data.yaml',   # 데이터셋 설정 파일 경로
        epochs=50,                          # 학습 epoch 수
        imgsz=640,                         # 입력 이미지 크기
        batch=16,                           # 배치 크기
        workers=4,                          # 데이터 로딩을 위한 병렬 작업 수
        project='/content/runs',           # 결과를 저장할 디렉토리
        name='exp'                           # 실험 이름
    )
```

2. YOLO v8 algorithm 실습

2) YOLO v8 기반 객체 인지 모델 학습

- Google Colab 환경 내 YOLO v8 Dataset 학습 [2 / 6]



→ YOLO v8 모델은 기존 YOLO 모델들과 같이

YOLOv8n → YOLOv8s → YOLOv8m → YOLOv8l → YOLOv8x 모델 순으로 구성

→ 모델 n에서 x로 갈수록 정확도는 향상되나, 모델이 무거워져 객체 인식 속도가 느림

2. YOLO v8 algorithm 실습

2) YOLO v8 기반 객체 인지 모델 학습

- Google Colab 환경 내 YOLO v8 Dataset 학습 [3 / 6]

Logging results to /content/runs/exp											
Starting training for 50 epochs...											
Epoch	GPU_mem	box_loss	cls_loss	df1_loss	Instances	Size					
1/50	2.45G	3.136	3.556	4.172	24	640: 100% ██████████ 55/55 [00:23<00:00, 2.31it/s] mAP50 mAP50-95): 100% ██████████ 6/6 [00:03<00:00, 1.73it/s]				all	187
2/50	2.25G	3.045	3.247	3.716	24	640: 100% ██████████ 55/55 [00:19<00:00, 2.78it/s] mAP50 mAP50-95): 100% ██████████ 6/6 [00:03<00:00, 1.85it/s]				all	187
3/50	2.26G	2.814	3.035	3.378	24	640: 100% ██████████ 55/55 [00:19<00:00, 2.89it/s] mAP50 mAP50-95): 100% ██████████ 6/6 [00:04<00:00, 1.45it/s]				all	187
4/50	2.26G	2.549	2.79	3.178	30	640: 100% ██████████ 55/55 [00:18<00:00, 3.00it/s] mAP50 mAP50-95): 100% ██████████ 6/6 [00:03<00:00, 1.62it/s]				all	187

→ 설정한 GPU를 기반으로 총 50 epoch를 반복하여 학습 수행

→ 1 epoch를 학습하는 데 약 20초 소요

2. YOLO v8 algorithm 실습

2) YOLO v8 기반 객체 인지 모델 학습

- Google Colab 환경 내 YOLO v8 Dataset 학습 [4 / 6]

→ 만약 GPU를 사용하지 않고, CPU를 사용하게 되면 학습 소요 시간 대폭 상승

Epoch	GPU_mem	box_loss	cls_loss	df_l_loss	Instances	Size
1/50	0G	3.131	3.583	4.174	31	640: 100% ██████████ 55/55 [14:15<00:00, 15.56s/it] mAP50 mAP50-95): 100% ██████████ 6/6 [01:09 00:00, 11.57s/it]
	Class	Images	Instances	Box(P)	R	

Epoch	GPU_mem	box_loss	cls_loss	df_l_loss	Instances	Size
2/50	0G	3.012	3.377	3.751	35	640: 100% ██████████ 55/55 [12:46<00:00, 13.94s/it] mAP50 mAP50-95): 100% ██████████ 6/6 [00:57<00:00, 9.63s/it]
	Class	Images	Instances	Box(P)	R	

CPU 기반 YOLO v8 학습

Epoch	GPU_mem	box_loss	cls_loss	df_l_loss	Instances	Size
1/50	2.59G	3.136	3.556	4.172	24	640: 100% ██████████ 55/55 [00:23<00:00, 2.33it/s] mAP50 mAP50-95): 100% ██████████ 6/6 [00:04<00:00, 1.37it/s]
	Class	Images	Instances	Box(P)	R	

Epoch	GPU_mem	box_loss	cls_loss	df_l_loss	Instances	Size
2/50	2.25G	3.045	3.247	3.716	24	640: 100% ██████████ 55/55 [00:20<00:00, 2.73it/s] mAP50 mAP50-95): 100% ██████████ 6/6 [00:04<00:00, 1.29it/s]
	Class	Images	Instances	Box(P)	R	

GPU 기반 YOLO v8 학습

2. YOLO v8 algorithm 실습

2) YOLO v8 기반 객체 인지 모델 학습

- Google Colab 환경 내 YOLO v8 Dataset 학습 [5 / 6]

```

Epoch    GPU_mem   box_loss   cls_loss   dfl_loss   Instances   Size
49/50    2.25G     0.6917     0.5002     1.265      11          640: 100%|██████████| 55/55 [00:18<00:00,  2.98it/s]
          Class      Images     Instances     Box(P)       R          mAP50  mAP50-95): 100%|██████████| 6/6 [00:02<00:00,  2.79it/s]           all     187

Epoch    GPU_mem   box_loss   cls_loss   dfl_loss   Instances   Size
50/50    2.25G     0.6795     0.4949     1.244      14          640: 100%|██████████| 55/55 [00:18<00:00,  3.01it/s]
          Class      Images     Instances     Box(P)       R          mAP50  mAP50-95): 100%|██████████| 6/6 [00:01<00:00,  3.20it/s]           all     187

50 epochs completed in 0.320 hours.
Optimizer stripped from /content/runs/exp/weights/last.pt, 6.2MB
Optimizer stripped from /content/runs/exp/weights/best.pt 6.2MB

Validating /content/runs/exp/weights/best.pt...
Ultralytics YOLOv8.2.57 🚀 Python-3.10.12 torch-2.3.0+cu121 CUDA:0 (Tesla T4, 15102MIB)
YOLOv8n summary (fused): 168 layers, 3,005,843 parameters, 0 gradients, 8.1 GFLOPs
          Class      Images     Instances     Box(P)       R          mAP50  mAP50-95): 100%|██████████| 6/6 [00:06<00:00,  1.06s/it]
          all     187     238     0.979     0.929     0.982     0.797
Speed: 1.0ms preprocess, 3.6ms inference, 0.0ms loss, 7.9ms postprocess per image
Results saved to /content/runs/exp
  
```

- ➔ 50 epoch를 학습하는 데 약 30분 가량 소요
- ➔ 학습 과정 중 특정 지표(대표적으로 'mAP')가 향상될 때마다 모델의 상태를 저장하며, 가장 좋은 성능을 보였을 때의 가중치를 'best.pt'로 저장
- ➔ 학습된 모델의 저장 경로는 '/content/runs/exp/weights/best.pt'

2. YOLO v8 algorithm 실습

2) YOLO v8 기반 객체 인지 모델 학습

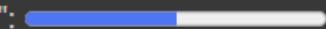
- Google Colab 환경 내 YOLO v8 Dataset 학습 [6 / 6]

```

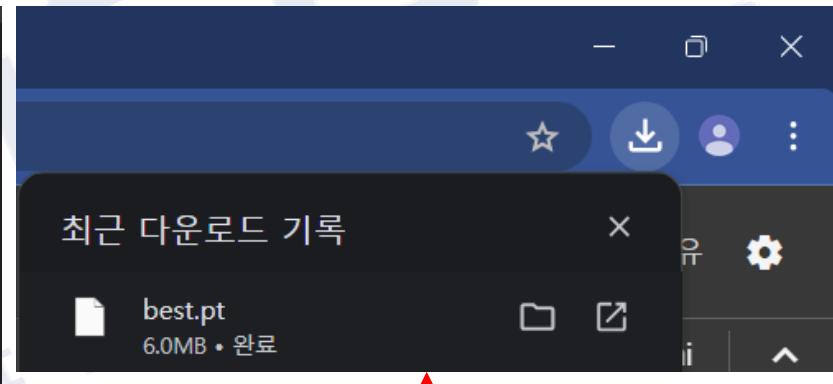
✓ # 학습된 모델인 'best.pt' 파일을 Local PC에 저장
from google.colab import files

# best.pt 파일 경로 (Google Drive에서의 경로)
model_path = '/content/runs/exp/weights/best.pt'

# 파일 다운로드
files.download(model_path)

⬇️ Downloading "best.pt": 

```



- 다음의 경로 '/content/runs/exp/weights/best.pt'에 저장되어 있는 학습된 모델을 Local PC에 저장
- 다운로드한 모델 'best.pt' 파일을 통해 어디서든 학습된 모델을 불러올 수 있음

2. YOLO v8 algorithm 실습

3) YOLO v8 기반 객체 인지 모델 검증

- Google Colab 환경 내 YOLO v8 학습 모델 검증 [1 / 6]

```
from ultralytics import YOLO
import matplotlib.pyplot as plt
import seaborn as sns
import cv2

# 학습된 모델 로드
model = YOLO('/content/runs/exp/weights/best.pt')

# 검증 데이터셋 설정 (예: coco128.yaml 파일)
validation_data = '/content/roboflow/data.yaml'

# 모델 검증
metrics = model.val(data=validation_data)
```

- 다음의 경로 '/content/runs/exp/weights/best.pt'에 저장되어 있는 학습된 모델을 load하여 data.yaml 파일에 위치한 validation data의 경로를 통해 validation 수행

2. YOLO v8 algorithm 실습

3) YOLO v8 기반 객체 인지 모델 검증

- Google Colab 환경 내 YOLO v8 학습 모델 검증 [2 / 6]

```
# 성능 지표 출력

# 'Precision' = 정확도
# => 'Precision'은 모델이 예측한 객체 중 실제로 올바르게 예측된 객체의 비율을 의미
precision = metrics.box.p

# 'Recall' = 재현율
# => 'Recall'은 실제 객체 중 모델이 올바르게 검출한 객체의 비율을 의미
recall = metrics.box.r

# 'mAP@0.5' = Mean Average Precision (IoU=0.5에서의 mAP)
# 'mAP@0.5:0.95' = Mean Average Precision (IoU=0.5에서 0.95까지의 mAP)
# => 'mAP'는 평균 정밀도를 의미하며, 모델이 얼마나 정확하게 다양한 객체를
# 탐지할 수 있는지를 평가하는 지표를 의미
# => 'mAP'는 객체 검출 모델의 성능을 평가하는 대표적인 지표로,
# 해당 값을 통해 모델의 성능을 판별할 수 있음. (높을수록 학습이 잘 된 모델)
# => 'mAP'는 'Precision'과 'Recall', 그리고 'AP(Average Precision)'의 평균값으로 도출
map_50 = metrics.box.map50
map_50_95 = metrics.box.map

print(f"Precision: {precision[0]}")
print(f"Recall: {recall[0]}")
print(f"mAP@0.5: {map_50}")
print(f"mAP@0.5:0.95: {map_50_95}")
```

→ 모델이 잘 학습되었는지를 평가하는 지표인 'Precision', 'Recall', 'mAP' 값을 출력

2. YOLO v8 algorithm 실습

3) YOLO v8 기반 객체 인지 모델 검증

- Google Colab 환경 내 YOLO v8 학습 모델 검증 [3 / 6]

```

Ultralytics YOLOv8.2.58 🚀 Python-3.10.12 torch-2.3.0+cu121 CUDA:0 (Tesla T4, 15102MiB)
YOLOv8n summary (fused): 168 layers, 3,005,843 parameters, 0 gradients, 8.1 GFLOPs
val: Scanning /content/roboflow/valid/labels.cache... 187 images, 0 backgrounds, 0 corrupt: 100%|██████████| 187/187 [00:00<?, ?it/s]
/usr/lib/python3.10/multiprocessing/popen_fork.py:66: RuntimeWarning: os.fork() was called. os.fork() is incompatible with multithreaded code
  self.pid = os.fork()
      Class      Images   Instances     Box(P       R      mAP50    mAP50-95): 100%|██████████| 12/12 [00:06<00:00,  1.93it/s]
      all        187       238      0.979     0.929     0.982     0.796
Speed: 0.9ms preprocess, 5.0ms inference, 0.0ms loss, 5.3ms postprocess per image
Results saved to runs/detect/val5
Precision: 0.9787486749626128
Recall: 0.9285714285714286
mAP@0.5: 0.982032413254485
mAP@0.5:0.95: 0.7958917054224899
  
```

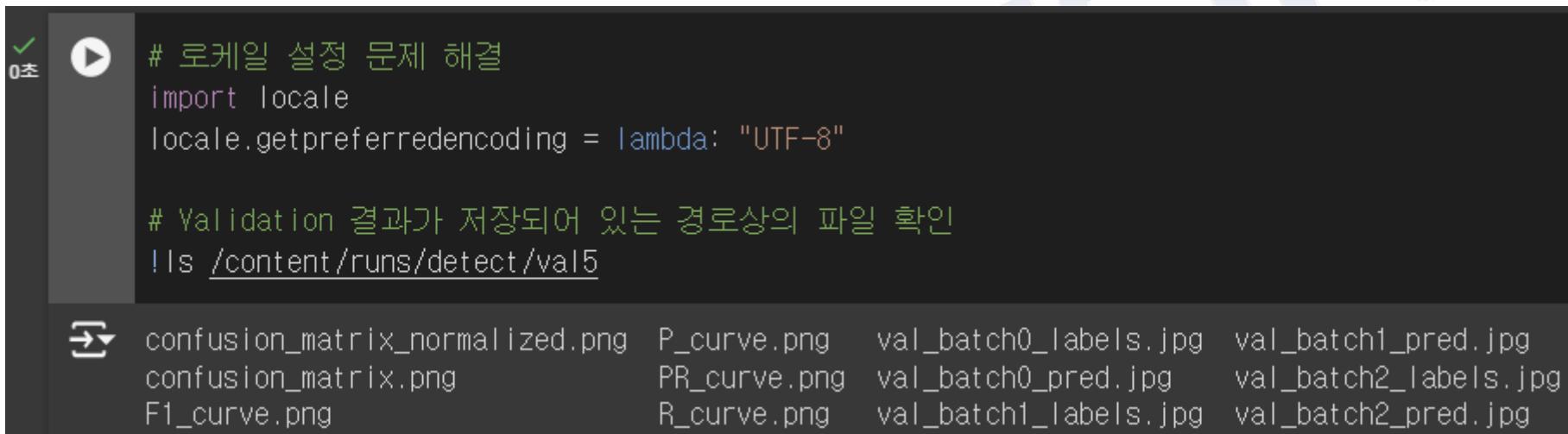
- Validation data를 기반으로 Validation을 수행한 뒤, 결과를 '/content/runs/detect/val5'에 저장
- 또한, 앞서 출력하였던 모델의 평가 지표인 'Precision', 'Recall', 'mAP' 값을 출력

평가 지표 설명 참고 → <https://ctkim.tistory.com/entry/mAPMean-Average-Precision-%EC%A0%95%EB%A6%AC>

2. YOLO v8 algorithm 실습

3) YOLO v8 기반 객체 인지 모델 검증

- Google Colab 환경 내 YOLO v8 학습 모델 검증 [4 / 6]



The screenshot shows a Google Colab notebook interface. On the left, there are two execution step indicators: a green checkmark for step 0초 and a play button icon for step 1초. The main area contains Python code for handling locale encoding and checking for validation files. Below the code, a file listing shows various validation-related files.

```
# 로케일 설정 문제 해결
import locale
locale.getpreferredencoding = lambda: "UTF-8"

# Validation 결과가 저장되어 있는 경로상의 파일 확인
!ls /content/runs/detect/val5
```

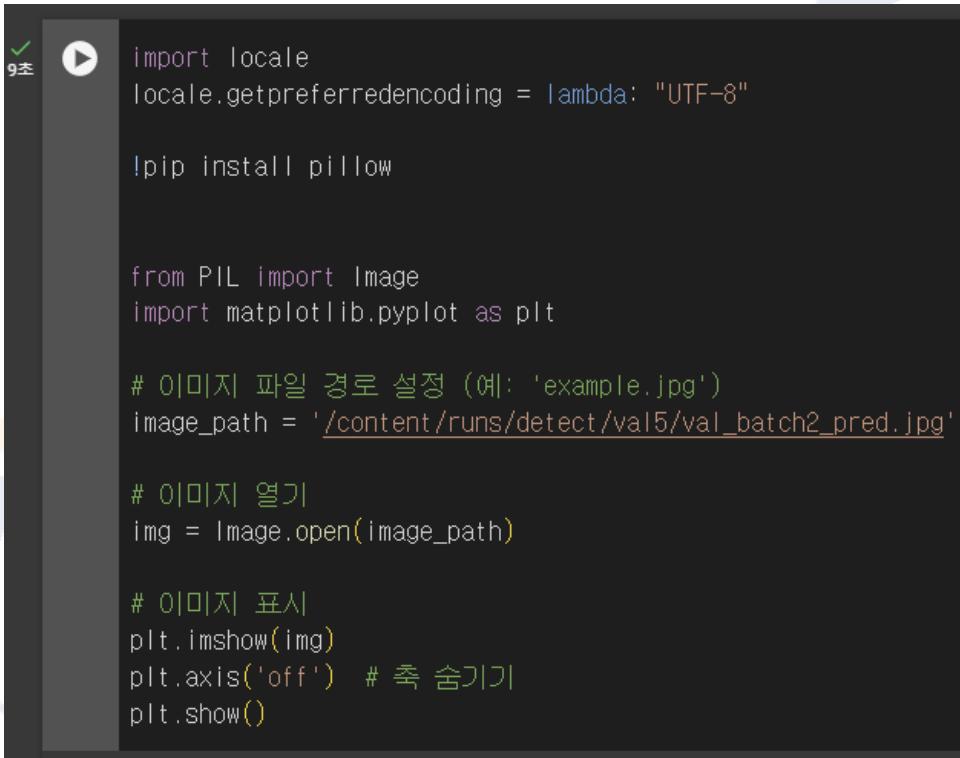
	confusion_matrix_normalized.png	P_curve.png	val_batch0_labels.jpg	val_batch1_pred.jpg
→	confusion_matrix.png	PR_curve.png	val_batch0_pred.jpg	val_batch2_labels.jpg
	F1_curve.png	R_curve.png	val_batch1_labels.jpg	val_batch2_pred.jpg

➔ Validation 결과가 저장되어 있는 경로상의 파일들 확인

2. YOLO v8 algorithm 실습

3) YOLO v8 기반 객체 인지 모델 검증

- Google Colab 환경 내 YOLO v8 학습 모델 검증 [5 / 6]



```
✓ 9초  play
import locale
locale.getpreferredencoding = lambda: "UTF-8"

!pip install pillow

from PIL import Image
import matplotlib.pyplot as plt

# 이미지 파일 경로 설정 (예: 'example.jpg')
image_path = '/content/runs/detect/val5/val_batch2_pred.jpg'

# 이미지 열기
img = Image.open(image_path)

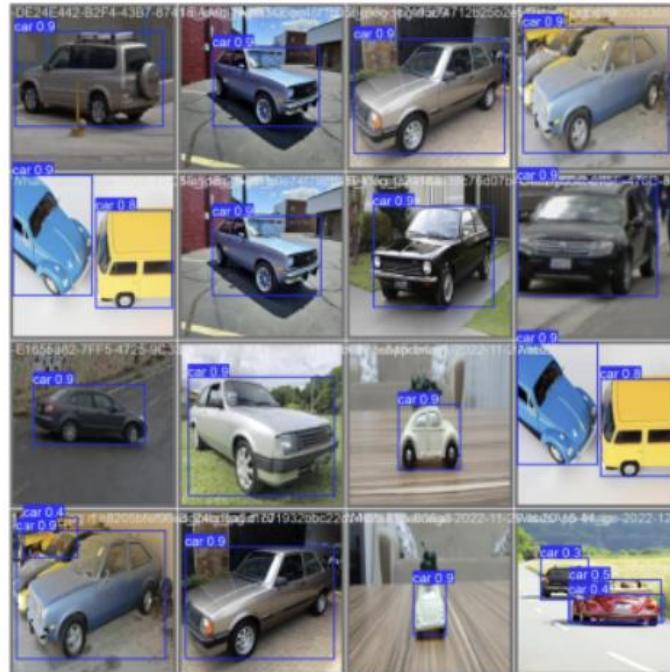
# 이미지 표시
plt.imshow(img)
plt.axis('off') # 축 숨기기
plt.show()
```

- ➔ Colab 환경에서 이미지를 화면에 출력하기 위해 'PIL(Python Imaging Library)'를 import 수행
- ➔ Validation 결과가 저장되어 있는 경로상의 파일 중, 본인이 원하는 파일의 경로를 작성

2. YOLO v8 algorithm 실습

3) YOLO v8 기반 객체 인지 모델 검증

- Google Colab 환경 내 YOLO v8 학습 모델 검증 [6 / 6]



- 다음의 Validation 결과 이미지가 화면에 출력
- 인식한 객체를 Bounding box로 표시하며, 객체의 Class와 Confidence Score를 함께 출력
(Confidence Score란, 모델이 예측한 객체가 특정 Class에 속할 확률을 의미)

2. YOLO v8 algorithm 실습

실습

✓ Custom Dataset 기반의 YOLO v8 학습 수행 실습

- 조건 1. 100 epoch 이상 학습 수행하기
- 조건 2. 'Car' Class는 제외한 다른 Class에 대한 학습 수행하기
- 조건 3. 1개의 Class가 아닌, 2개 이상의 Class를 바탕으로 학습 수행하기



3. Github Repository

3. Github Repository

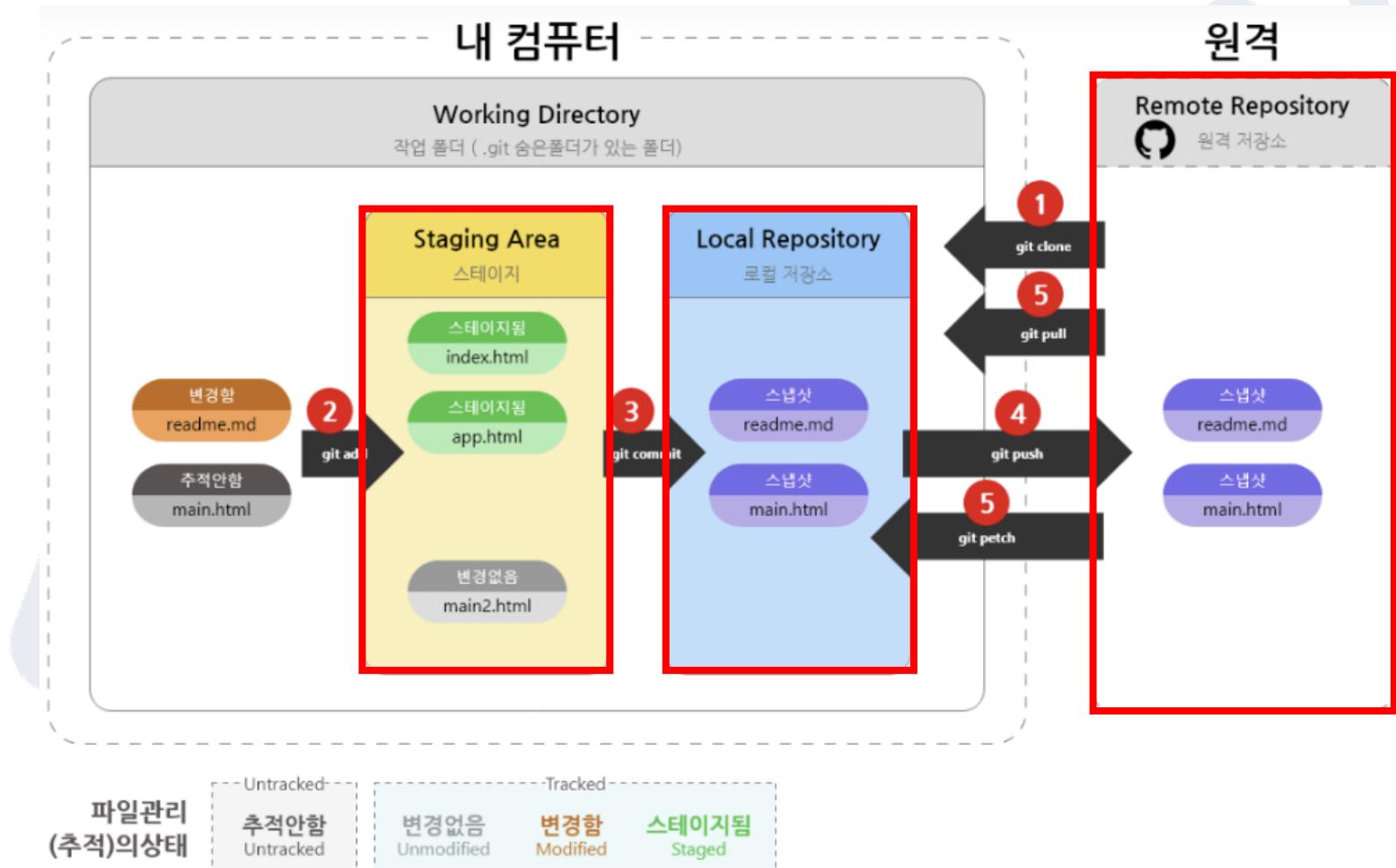
- Github란?



- Github란, 소프트웨어 개발 프로젝트를 관리하고 버전 관리를 할 수 있는 웹 기반 플랫폼
- 2008년 ‘톰 프레스턴워너’와 ‘크리스 원스트래스’ 외 다수의 개발자들에 의해 개발됨
- 버전 관리 / 협업 도구 / 프로젝트 관리 / 코드 호스팅 / 자동화 및 통합 / 소셜 코딩 / 문서화 등 다양한 기능을 지원
- Github는 소프트웨어 개발의 모든 단계를 지원하며, 개발자들이 보다 효율적으로 협업하고 프로젝트를 관리할 수 있도록 도와주는 강력한 도구

3. Github Repository

● Github 구조



Staging Area → Local Repository → Remote Repository

3. Github Repository

- Staging Area (스테이지)

- commit(커밋)전에 파일의 snapshot(스냅샷)을 저장하는 임시 영역
 - snapshot이란, 특정 시점의 파일이나 시스템 상태를 기록한 것
 - snapshot을 통해 특정 시점의 프로젝트 상태를 정확히 복원 가능
- 작업중인 directory(디렉토리)에서 변경된 파일들을 추가
- 변경 사항을 commit하기 전에 어떤 파일과 변경 사항이 포함될지 준비 가능
- ‘git add’ 명령어를 사용하여 파일을 Staging Area에 추가
 - `$ git add <file>` # 특정 파일을 Staging Area에 추가
 - `$ git add .` # 모든 변경된 파일을 Staging Area에 추가
- Staging Area는 commit할 준비가 된 파일들의 목록을 가지고 있음

3. Github Repository

● Local Repository (로컬 저장소)

- Local Repository는 사용자의 로컬 컴퓨터에 저장된 Git 저장소를 의미
- Local Repository에는 프로젝트의 모든 commit history와 branch 정보가 포함되어 있음
- Local Repository에는 로컬에서 commit된 모든 변경 사항이 저장됨
- 또한, 로컬에서 버전 관리를 수행할 수 있으며, 인터넷 연결 없이도 작업이 가능
- ‘git commit’ 명령어를 사용하여 Staging Area의 파일을 Local Repository에 commit 가능
 - ➔ `$ git commit -m “Commit message”` # Staging Area의 파일을 Local Repository에 commit
 - ➔ `$ git log` # commit history 확인

3. Github Repository

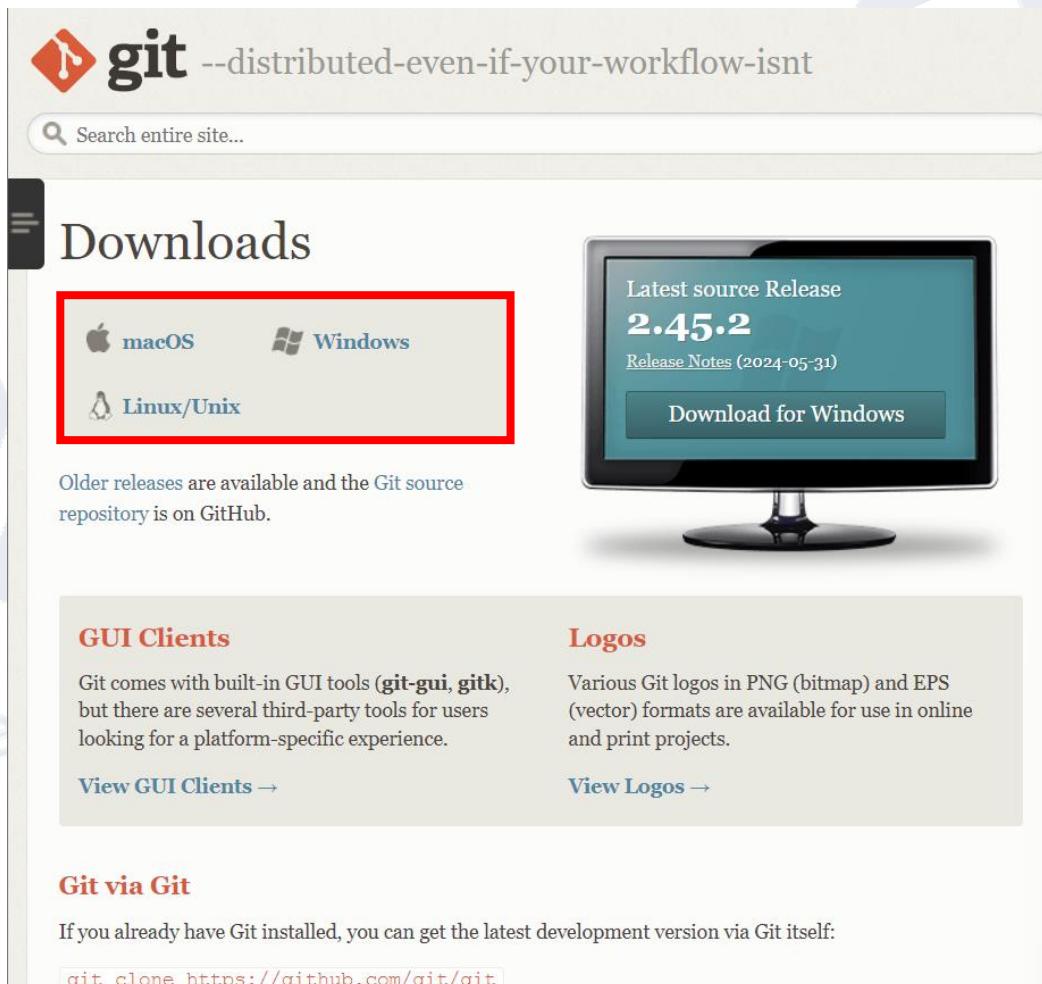
● Remote Repository (원격 저장소)

- **Remote Repository**는 원격 서버에 저장된 Git 저장소로, 원격 서버에 저장된 코드와 변경 사항을 공유하고 동기화 수행
 - **Remote Repository**는 팀원들과 협업할 때 공통으로 사용하는 중앙 저장소
 - ‘git push’ 명령어를 사용하여 **Local Repository**의 변경 사항을 **Remote Repository**로 push 가능
 - ‘git pull’ 또는 ‘git fetch’ 명령어를 통해 **Remote Repository**의 변경 사항을 Local로 가져올 수 있음
- **\$ git push origin master** # Local Repository의 master branch를 Remote Repository로 push
- **\$ git pull origin master** # Remote Repository의 master branch의 변경 사항을 Local로 가져옴
- **\$ git fetch origin** # Remote Repository의 변경 사항을 Local로 가져오지만, 자동 병합은 X

3. Github Repository

● Git 설치

- <https://git-scm.com/downloads> → 해당 링크를 통해 본인의 OS에 맞는 Git을 다운로드 !!



3. Github Repository

- Git 설정

- Git 설치 후, 가장 먼저 “사용자 이름”과 “이메일 주소”를 설정해야 함

→ `$ git config --global user.name "Hwasu Lee"`

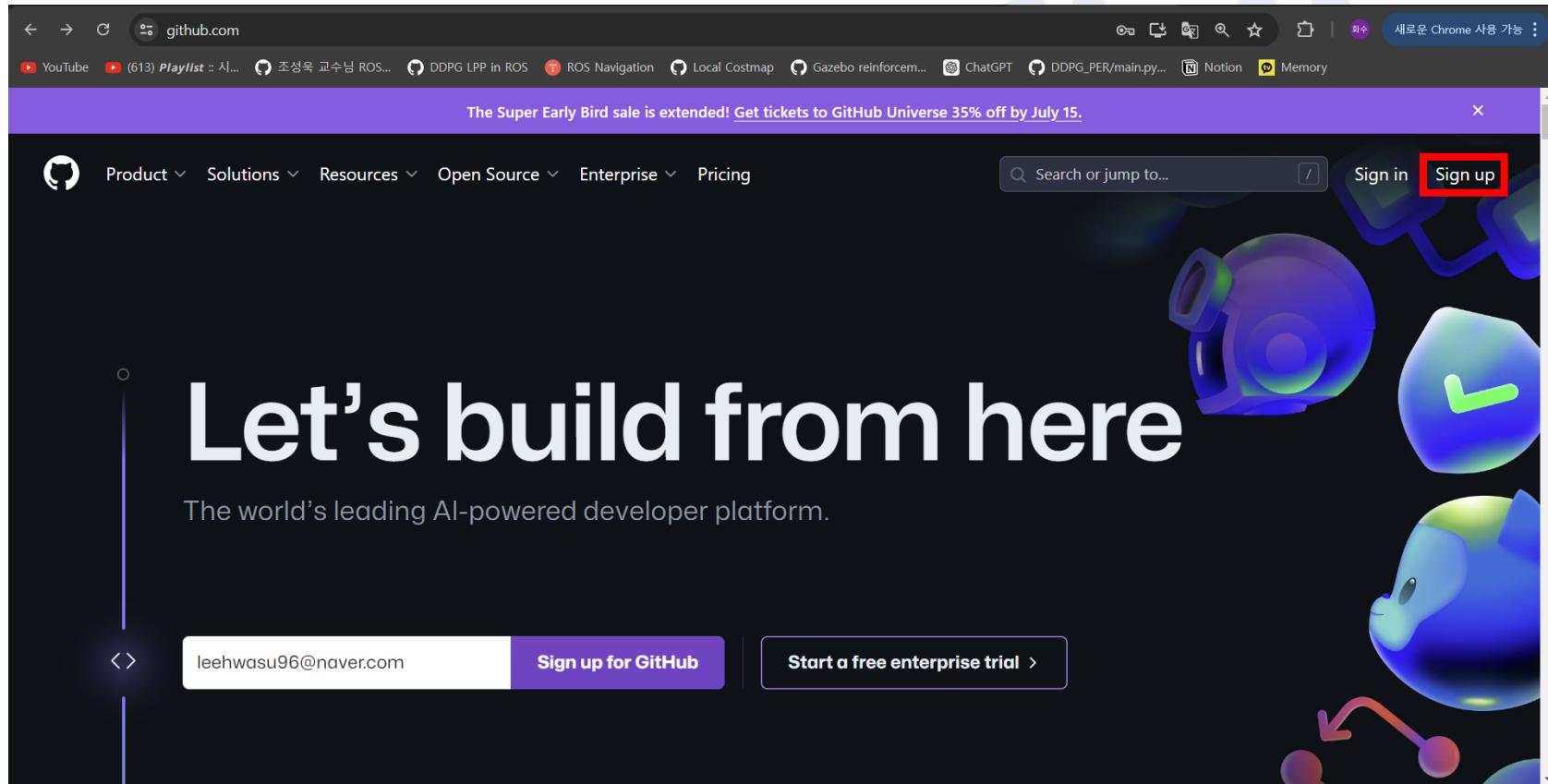
→ `$ git config --global user.email leehwasu96@naver.com`

- “--global” 옵션 설정은 최초 1회만 수행
 - 만약 각 프로젝트별로 다른 이름과 이메일 주소를 사용하고 싶은 경우, “--global” 옵션을 빼고 명령을 실행

3. Github Repository

● Github 계정 생성

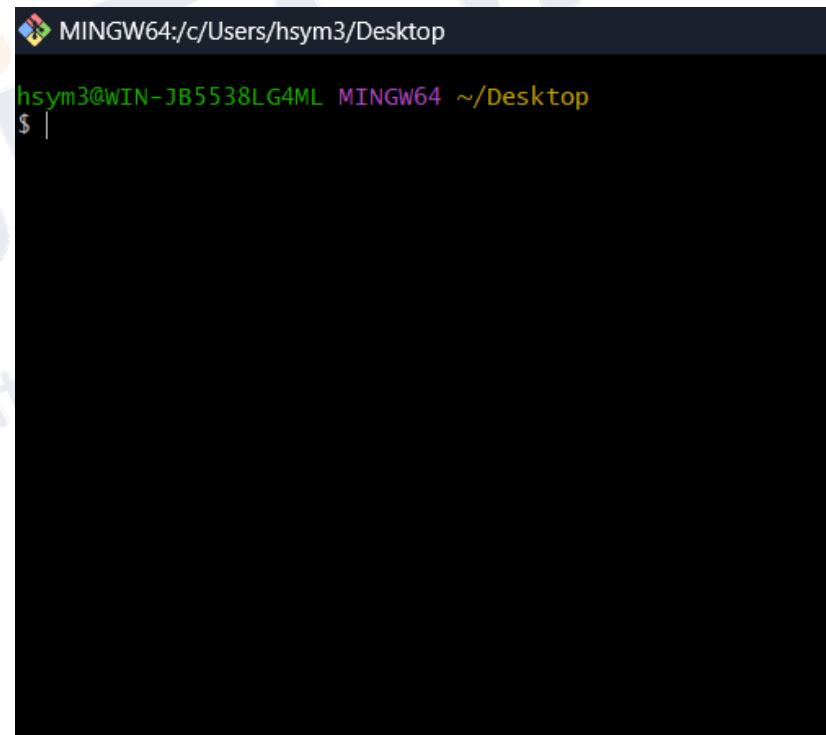
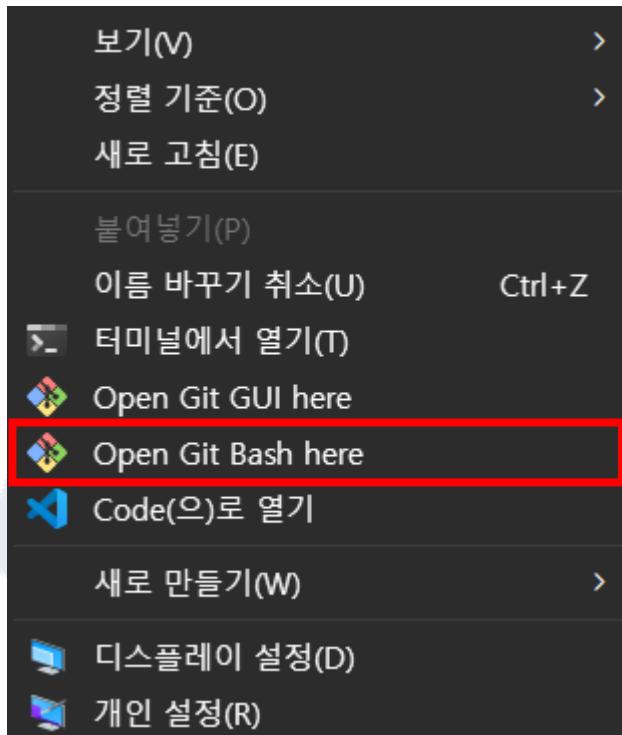
- <https://github.com/> → 해당 링크를 통해 Github 홈페이지로 이동 후, “Sign up”을 통해 계정 생성



3. Github Repository

● Github Local Repository(로컬 저장소) 생성 및 설정

- 바탕화면에 빈 공간에 마우스 우 클릭 후, “Open Git Bash here”을 클릭하여 터미널 생성



```
MINGW64:/c/Users/hsym3/Desktop
hsym3@WIN-JB5538LG4ML MINGW64 ~/Desktop
$ |
```

3. Github Repository

● Github Local Repository(로컬 저장소) 생성 및 설정

- 터미널 생성 후, 다음의 명령어를 실행하여 본인의 Working directory로 이동

```
MINGW64:/c/Users/hsym3/Desktop/My_github_code

hsym3@WIN-JB5538LG4ML MINGW64 ~/Desktop
$ cd My_github_code/

hsym3@WIN-JB5538LG4ML MINGW64 ~/Desktop/My_github_code
$ git init
Initialized empty Git repository in C:/Users/hsym3/Desktop/My_github_code/.git/
hsym3@WIN-JB5538LG4ML MINGW64 ~/Desktop/My_github_code (master)
$
```

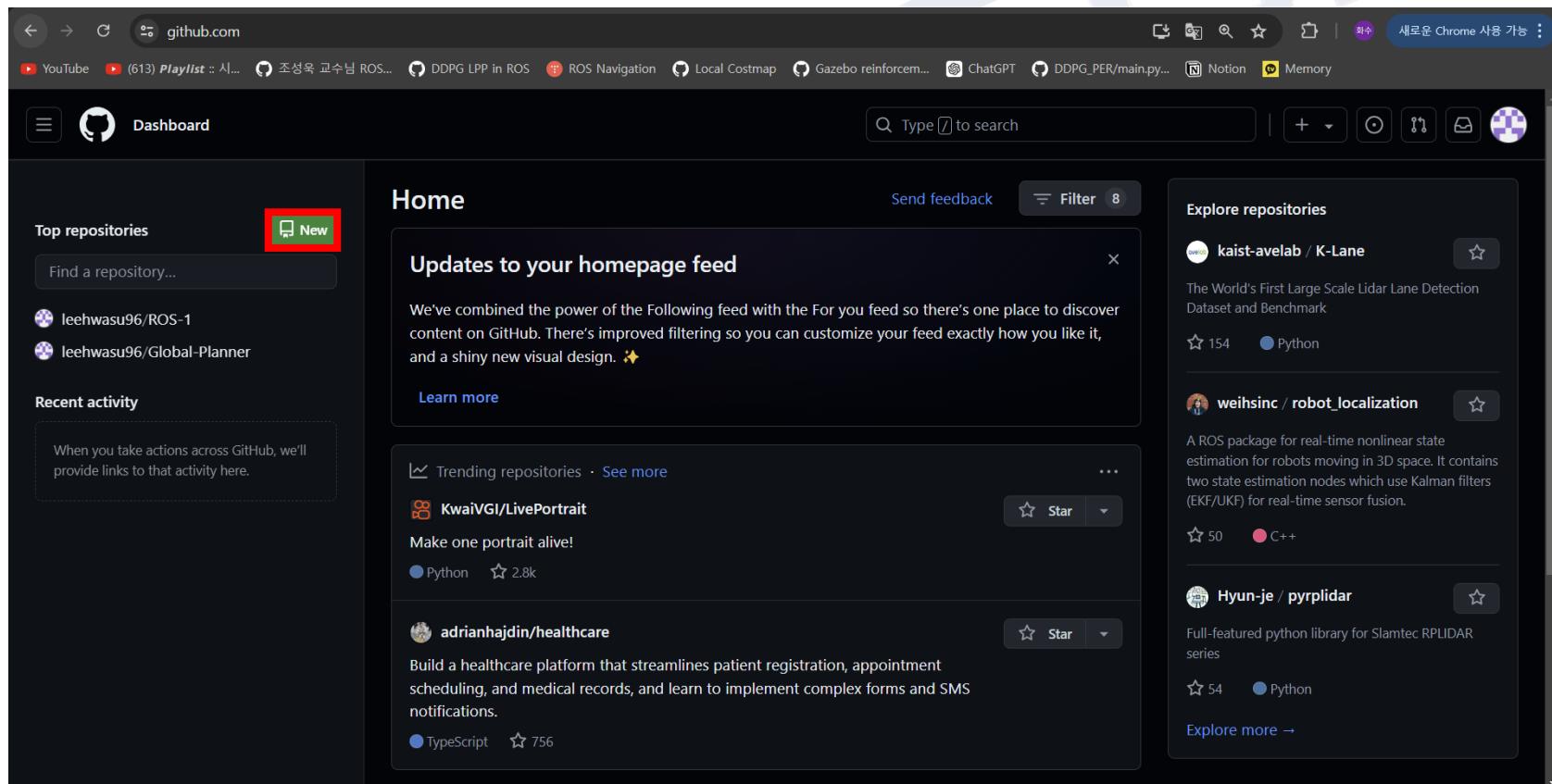
\$ cd My_github_code/ → Github에 업로드하기 위한 Working Directory로 이동

\$ git init → 일반적인 directory를 git working tree로 변환

3. Github Repository

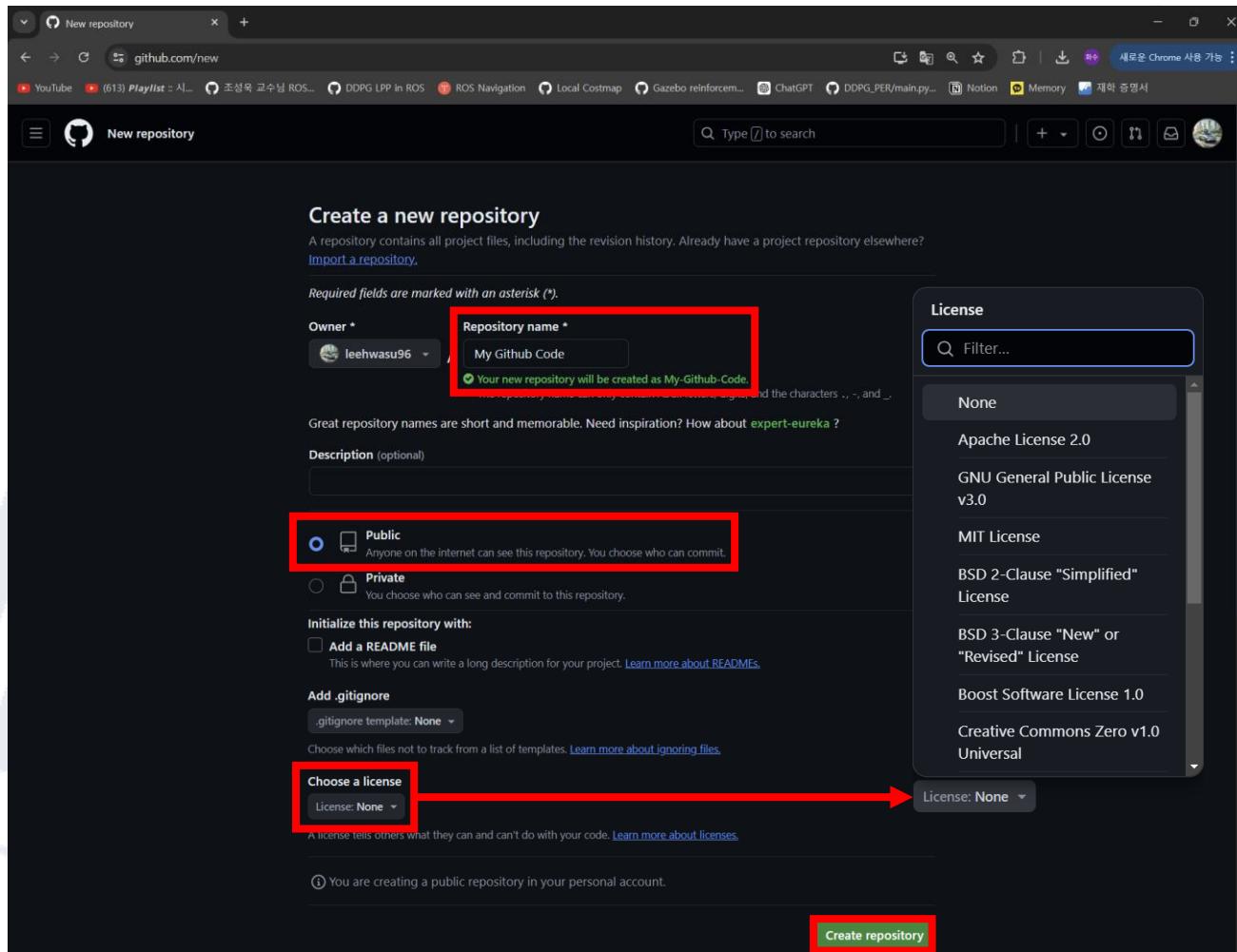
● Github Remote Repository(원격 저장소) 생성 및 설정

- “Sign up”을 통해 계정 생성 후, 아래 그림과 같이 “New”를 클릭하여 새로운 Repository를 생성
→ Github에 login 되어 있는 상태여야 함 !!



3. Github Repository

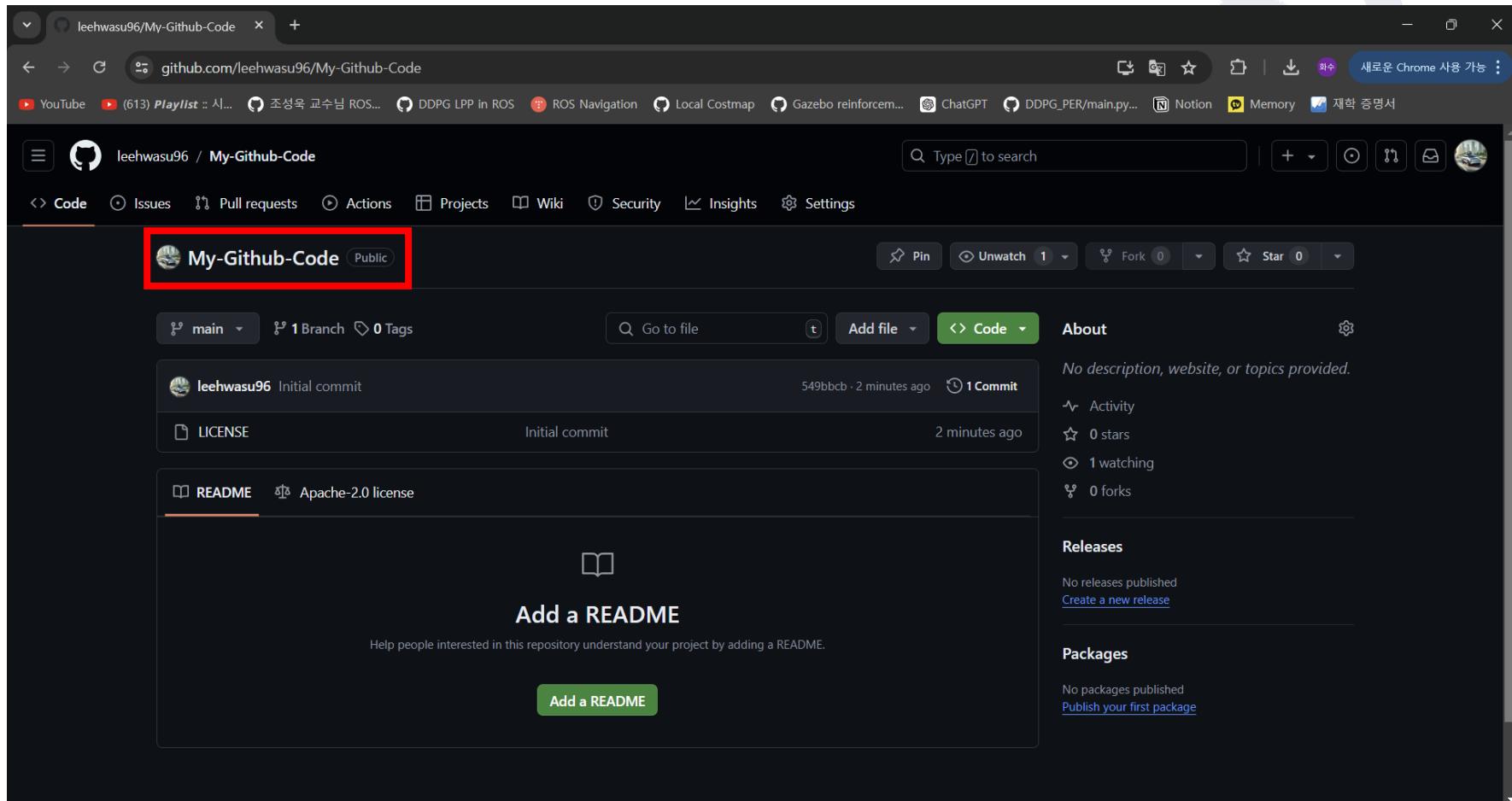
● Github Remote Repository(원격 저장소) 생성 및 설정



Repository name → Public/Private → Choose a license → Create repository

3. Github Repository

● Github Remote Repository(원격 저장소) 생성 및 설정



→ 위 그림과 같이 Github repository가 생성되면 준비 완료

3. Github Repository

● Github Local & Remote Repository 연동

```
hsym3@WIN-JB5538LG4ML MINGW64 ~/Desktop/My_github_code (master)
$ git remote add origin https://github.com/leehwasu96/My-Github-Code.git

hsym3@WIN-JB5538LG4ML MINGW64 ~/Desktop/My_github_code (master)
$ git checkout -b main
Switched to a new branch 'main'
```

\$ git remote add <remote-name> <git address>

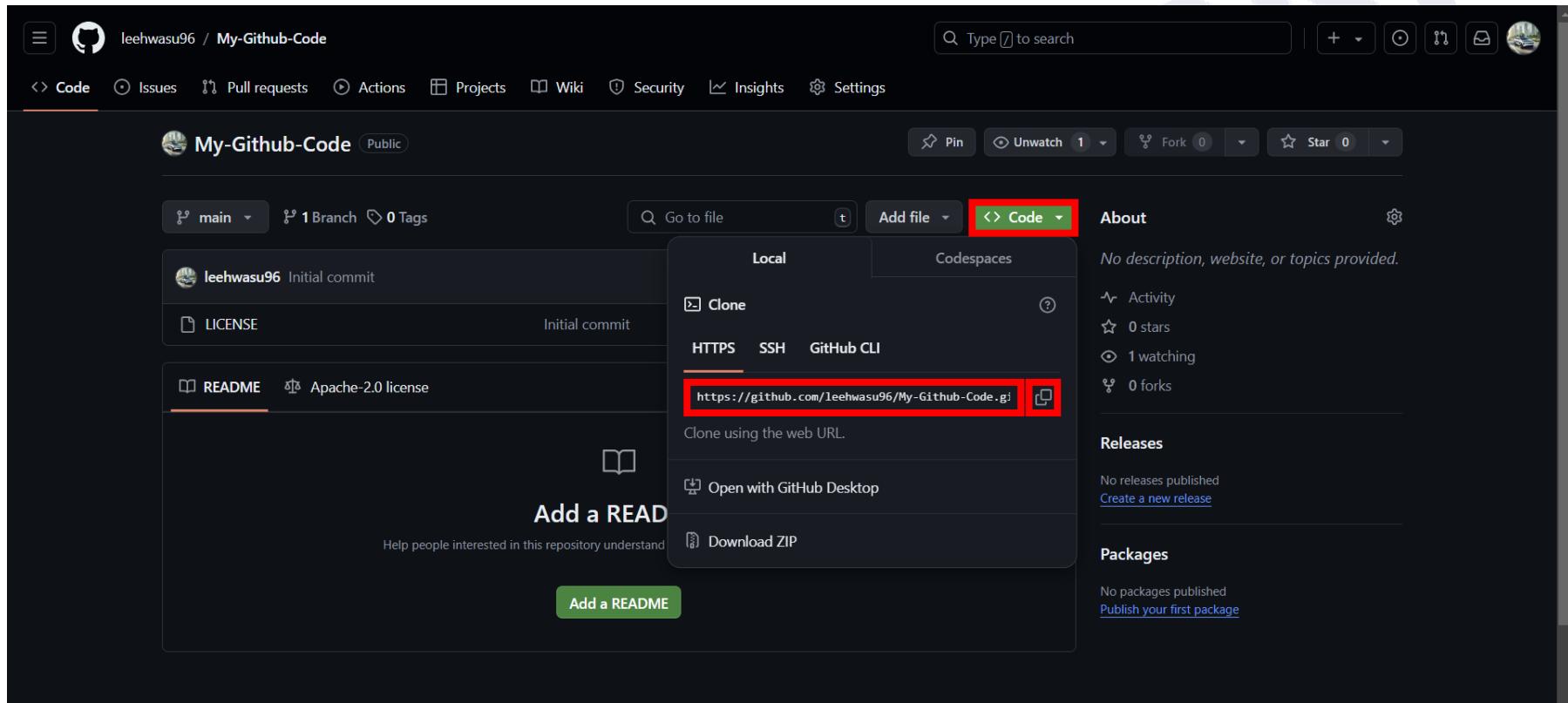
- ➔ remote-name : Remote Repository에 대한 별칭 (별칭으로 'origin'을 보편적으로 사용)
- ➔ git address : 생성한 Remote Repository(원격 저장소)의 URL
- ➔ Remote Repository의 URL 대신, 별칭 'origin'을 통해 간단히 명령어 작성 가능

\$ git checkout -b main

- ➔ 기본 branch를 'main'으로 설정한 뒤, 해당 branch로 이동
- ➔ 최신 Git 버전에서는 기본 branch가 'main'이지만, 이전 버전에서는 'master'가 기본 branch일 수 있음

3. Github Repository

● Github Local & Remote Repository 연동

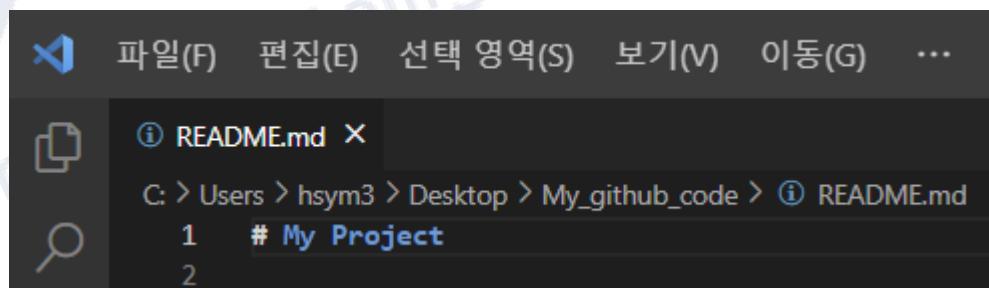
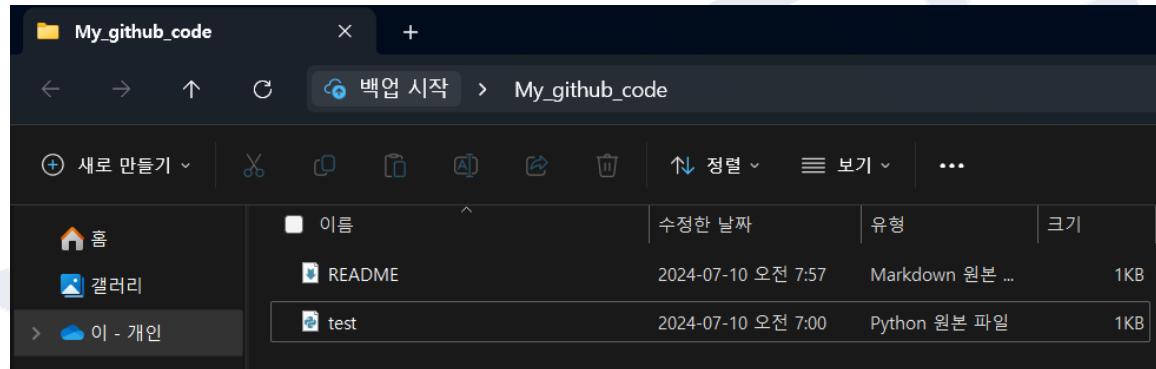


➔ 생성한 Remote Repository(원격 저장소)의 URL은 다음 과정을 통해 복사 가능

3. Github Repository

● Staging Area 파일 업로드

```
hsym3@WIN-JB5538LG4ML MINGW64 ~/Desktop/My_github_code (main)
$ echo "# My Project" > README.md
```



\$ echo "# My Project" > README.md

➔ README.md 파일 생성 후, 해당 파일에 "# My Project" 작성

3. Github Repository

● Staging Area 파일 업로드

```
hsym3@WIN-JB5538LG4ML MINGW64 ~/Desktop/My_github_code (main)
$ git add README.md
warning: in the working copy of 'README.md', LF will be replaced by CRLF the next time Git touches it

hsym3@WIN-JB5538LG4ML MINGW64 ~/Desktop/My_github_code (main)
$ git add test.py

hsym3@WIN-JB5538LG4ML MINGW64 ~/Desktop/My_github_code (main)
$ git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
    new file:   test.py
```

\$ git add <file name>

→ 특정 파일(file name)을 Staging Area에 업로드

\$ git status

→ Staging Area에 파일이 업로드 되었는지 확인

3. Github Repository

- Staging Area 파일 업로드

\$ git add . (또는 \$ git add --all)

→ Staging Area에 한 번에 업로드

\$ git add /<폴더 이름>/*

→ 특정 폴더에 위치한 모든 파일을 Staging Area에 업로드

\$ git add *.py

→ Working Directory에 위치한 Python 파일만을 업로드

3. Github Repository

- Local Repository에 파일 Commit

```
hsym3@WIN-JB5538LG4ML MINGW64 ~/Desktop/My_github_code (main)
$ git commit -m "Initial commit"
[main (root-commit) d736337] Initial commit
 1 file changed, 1 insertion(+)
 create mode 100644 README.md

hsym3@WIN-JB5538LG4ML MINGW64 ~/Desktop/My_github_code (main)
$ git commit -m "Python Test File"
[main e26e8b7] Python Test File
 1 file changed, 1 insertion(+)
 create mode 100644 test.py
```

\$ git commit -m “Commit시, 작성하고자 하는 메시지”

→ Staging Area에 업로드한 파일을 Commit하여 **Local Repository**로 업로드

→ -m + “작성하고자 하는 메시지”

해당 예시에서는 “Initial commit”과 “Python Test File”이라는 메시지와 함께 업로드 됨

3. Github Repository

- Local Repository에 파일 Commit

```
hsym3@WIN-JB5538LG4ML MINGW64 ~/Desktop/My_github_code (master)
$ git commit --amend
```



```
MINGW64:/c/Users/hsym3/Desktop/My_github_code
First commit

# Please enter the commit message for your changes. Lines starting
# with '#' will be ignored, and an empty message aborts the commit.
#
# Date:      Wed Jul 10 07:10:23 2024 +0900
#
# On branch master
#
# Initial commit
#
# Changes to be committed:
#       new file:   test.py
```



```
hsym3@WIN-JB5538LG4ML MINGW64 ~/Desktop/My_github_code (master)
$ git commit --amend
[master 46c8959] First commit
Date: Wed Jul 10 07:10:23 2024 +0900
1 file changed, 1 insertion(+)
create mode 100644 test.py
```

만약 Commit 메시지를
수정하고 싶다면,

\$ git commit --amend

→ ‘i’를 눌러 수정 시작

→ 수정이 끝나면

‘ESC’ + ‘:wq’ + ‘Enter’를

눌러 수정한 내용 저장

→ 수정 완료

3. Github Repository

- Remote Repository에 파일 Push

```
hsym3@WIN-JB5538LG4ML MINGW64 ~/Desktop/My_github_code (main)
$ git branch
* main

hsym3@WIN-JB5538LG4ML MINGW64 ~/Desktop/My_github_code (main)
$ git push origin main
Enumerating objects: 9, done.
Counting objects: 100% (9/9), done.
Delta compression using up to 16 threads
Compressing objects: 100% (5/5), done.
Writing objects: 100% (8/8), 810 bytes | 810.00 KiB/s, done.
Total 8 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To https://github.com/leehwasu96/My-Github-Code.git
  a639acf..0965ab8 main -> main
```

\$ git branch

→ 현재 어떤 branch에 있는지 확인

\$ git push <remote-name> <branch-name>

→ remote-name: 앞서 정의한 원격 저장소 별칭

→ branch-name: 현재 위치한 branch 이름

3. Github Repository

● Remote Repository에 파일 Push

→ 만약 Push 과정에서 아래와 같은 오류가 발생한다면

```
hsym3@WIN-JB5538LG4ML MINGW64 ~/Desktop/My_github_code (main)
$ git push origin main
To https://github.com/leehwasu96/My-Github-Code.git
 ! [rejected]      main -> main (non-fast-forward)
error: failed to push some refs to 'https://github.com/leehwasu96/My-Github-Code.git'
hint: Updates were rejected because the tip of your current branch is behind
hint: its remote counterpart. If you want to integrate the remote changes,
hint: use 'git pull' before pushing again.
hint: See the 'Note about fast-forwards' in 'git push --help' for details.
```

→ 다음과 같이 해결

```
hsym3@WIN-JB5538LG4ML MINGW64 ~/Desktop/My_github_code (main)
$ git pull origin main --allow-unrelated-histories
From https://github.com/leehwasu96/My-Github-Code
 * branch            main      -> FETCH_HEAD
Merge made by the 'ort' strategy.
 LICENSE | 201 ++++++
 1 file changed, 201 insertions(+)
 create mode 100644 LICENSE
```

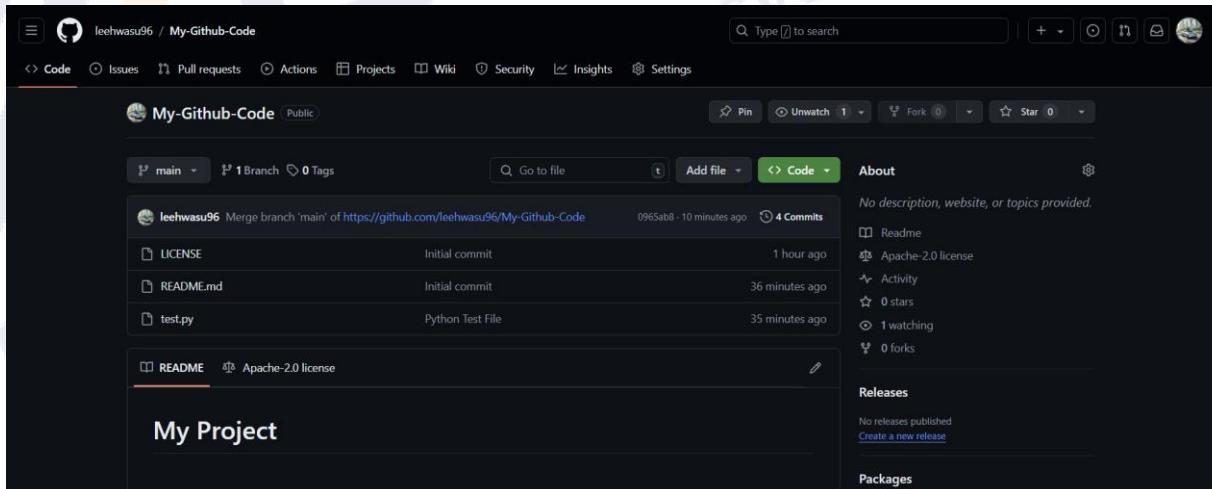
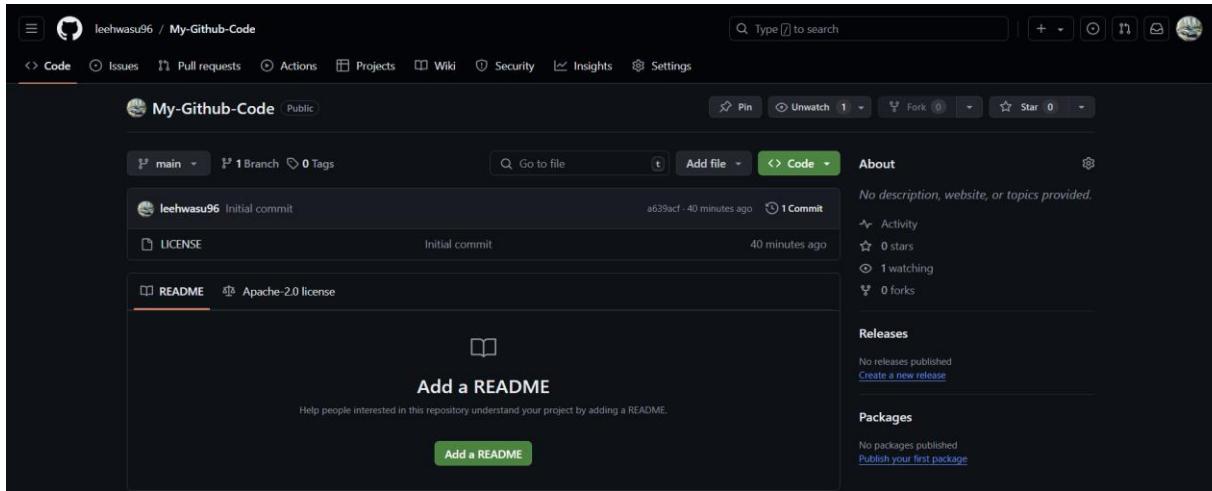
→ 이후, 다시
Push 진행

```
MINGW64:/c/Users/hsym3/Desktop/My_github_code
Merge branch 'main' of https://github.com/leehwasu96/My-Github-Code
# Please enter a commit message to explain why this merge is necessary,
# especially if it merges an updated upstream into a topic branch.
#
# Lines starting with '#' will be ignored, and an empty message aborts
# the commit.
~
```

→ ‘ESC’ + ‘:wq’ + ‘Enter’를
눌러 내용 저장

3. Github Repository

- Remote Repository에 파일 Push



➔ Local Repository에서 Remote Repository로 파일 업로드 완료 !!

3. Github Repository

- Remote Repository의 수정사항을 Local Repository로 Pull

```
hsym3@WIN-JB5538LG4ML MINGW64 ~/Desktop/My_github_code (main)
$ git pull origin main
From https://github.com/leehwasu96/My-Github-Code
 * branch           main      -> FETCH_HEAD
Already up to date.
```

\$ git pull <remote-name> <branch-name>

- ➔ remote-name: 앞서 정의한 원격 저장소 별칭
- ➔ branch-name: 현재 위치한 branch 이름
- ➔ Remote Repository의 특정 branch에서 수정사항을 Local Repository로 가져오기
- ➔ Remote Repository의 특정 branch의 수정사항을 가져와 현재 Local branch에 병합
- ➔ 위 예시에서는 수정사항이 없기 때문에 'Already up to date.'라는 문장이 출력됨

3. Github Repository

- Remote Repository의 수정사항을 Local Repository로 Fetch

```
hsym3@WIN-JB5538LG4ML MINGW64 ~/Desktop/My_github_code (main)
$ git fetch origin
```

\$ git fetch <remote-name>

- ➔ remote-name: 앞서 정의한 원격 저장소 별칭
- ➔ Remote Repository의 특정 branch에서 수정사항을 Local Repository로 가져오기
- ➔ Remote Repository의 수정사항을 가져오지만, Local branch에는 자동으로 병합 X
- ➔ 따라서, Remote Repository의 수정사항이 있는지 확인만 하고 변경된 데이터를 Local branch에 자동 병합은 수행하지 않으므로, Fetch 수행 후 Pull 실행 시 더욱 안전

2. Github Repository Setup

- Github 기반 코드 업로드 과정

- ① Local Repository(로컬 저장소) 및 Remote Repository(원격 저장소) 생성
- ② Local Repository와 Remote Repository 연동
- ③ Staging Area에 파일 업로드
- ④ 업로드한 파일을 Commit하여 Local Repository에 업로드
- ⑤ Commit한 파일을 Push하여 Remote Repository에 업로드
- ⑥ Remote Repository의 수정사항을 Pull하여 Local Repository로 업로드

QnA

E-mail : leehwasu9696@inu.ac.kr

Mobile : 010-8864-5585



청주대학교
미래형자동차 인력양성사업단

자율주행 분야

단기집중 교육과정



6월 11일(화)
오후 4시~8시

리눅스 설치,
ROS 설치 및 ROS 개발환경 구축

융합관
410호

*개인 노트북 필수 지참(대여불가)

온라인
사전교육

Python 교육(ROS 기반)



새천년종합정보관
107A호
경진대회 참여학생

7월 2일~
5일(화~금)
오후 1시~6시

라이다 센서 교육

7월 8일~
10일(월~수)
오후 1시~6시

ROS 활용 교육(Python 기반)

융합관
410호

*개인 노트북 필수 지참(대여불가)

7월 15일~
17일(월~수)
오후 1시~6시

Python OpenCV 설치 및 응용

새천년종합정보관
409호
*개인 노트북 필수 지참(대여불가)



청주대학교
CHEONGJU UNIVERSITY

Thank you.