
# Nvidia AI Specialist Certification

## Title: Confirmation of wearing or not wearing a helmet at work site in the work place

### Overview of the Project

- **Opening background information**

Many apartments, people, friends, and the people who built the houses I live in in Korea are thanks to the daily workers who work in labor and construction sites. Thanks to the infrastructure that construction workers sweat and have worked with their hands, we (Korea) are living in a convenient place 365 days a year. To prevent accidents caused by not wearing helmets during workers' work, it helps prevent accidents by detecting helmets or not wearing them in real time.

- **General Description of the Current Project**

The project utilizes (Yolov5) to detect helmet wearing and not wearing by construction site workers in real time, inform them of not wearing them, and help reduce the probability of accidents.

- **Proposed Idea for enhancements to the project**

1. Real-time processing: Real-time detection with Yolov5

2. Stability: Can distinguish between non-helmet wearers and wearers in real time to prevent this and reduce accidents

- **Value and significance of this project**

https://i.namu.wiki/i/RaVqvlgicLbAwlxUnbtQLqCipcEVNuLp4mutPLocYhQTrU_YyUEeGfDX7DMkD_aI_3OgdLCksqX9S-AJnW_HiA.mp4

https://i.namu.wiki/i/P-GqiVDiQB4jjLZoMSkHVcI6O9C3oMam86HNbmI_xhlYSdgViA8v56bbsYtR6hkOLAC0R3ylOptZqf6dhsnDmg.mp4

There are various accidents at the labor site, but there are many accidents that occur due to not wearing a helmet. In fact, there are many accidents that would not have occurred if a helmet was worn. (**Video that tells you the risk of not wearing a helmet)**Through this project, this project will reduce the helmet of workers, and the possibility of accidents will also reduce the possibility of accidents.This can be applied to the future, and it seems that there is also possible to apply in future**.**

- **Current limitations**

Although it is possible to distinguish between not wearing a helmet and wearing it, it is necessary to improve the performance of distinguishing between not wearing a helmet and wearing a hat or something on the head

- **Literature review**

Learning diverse training data and AI for helmets: accurate image recognition, accurate understanding of guest house detection and classification

---

## Image Acquisition Method:

- **I got the video from YouTube because I can't actually go into the labor field and film it**

https://drive.google.com/drive/folders/1TTbMUXGdHJemgqlmActXrOkjXf1iSfc2?usp=sharing

## Learning data extraction and Learning annotation:

**To learn from Yolov5 to 640 resolution images, the image is made into 640 x 640.**

### Video resolution adjustment

비디오 리사이저 - 온라인에서 무료로 비디오 해상도 변경
모든 비디오에 무료로 온라인 비디오 크기 조정기를 사용하십시오! MP4, AVI, 3GP 등 모든 비디오 형식을 지원합니다! 최대 4 GB의 파일에 대한 비디오 해상도를 변경할 수 있습니다! 지금 무료 도구를 사용해 보세요!
https://online-video-cutter.com/ko/resize-video

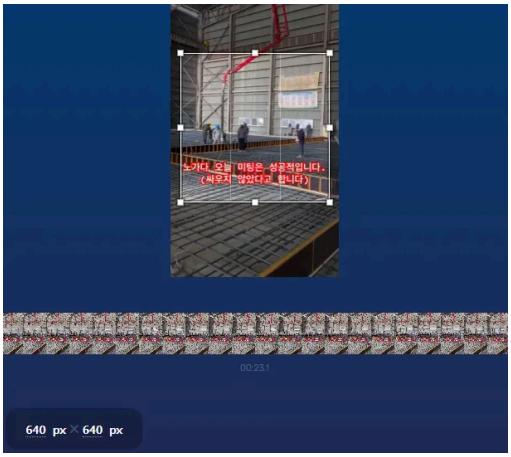
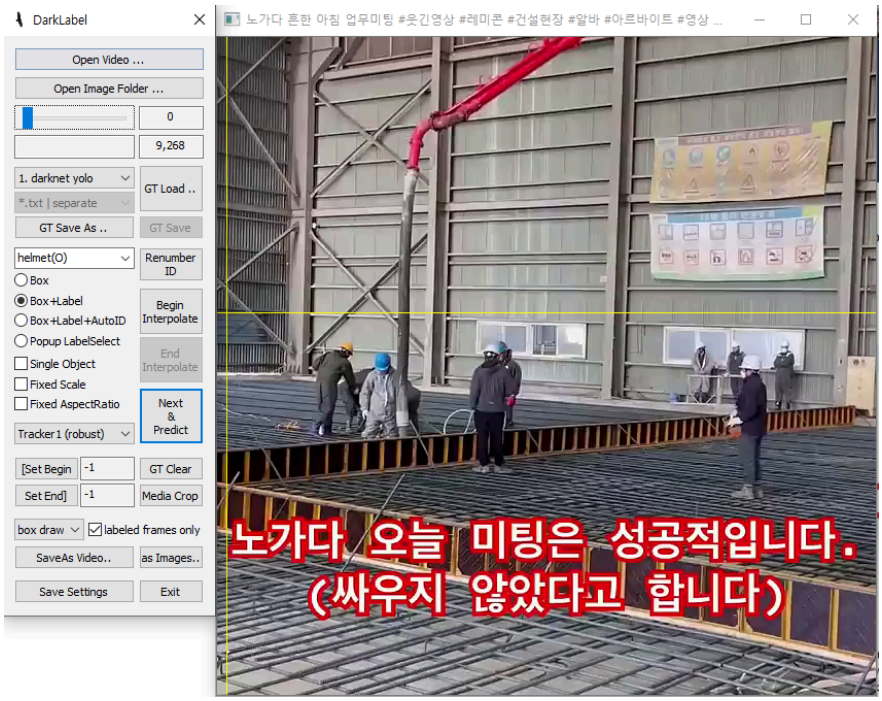



**Image conversion is performed on a frame-by-frame basis using the DarkLabel program.**

| | | | |
|---|---|---|---|
| DarkLabel | 2021-09-04 오후 6:12 | 응용 프로그램 | 20,261KB |
| darklabel | 2024-11-21 오전 4:16 | SAV 파일 | 1KB |
| darklabel | 2024-11-21 오전 3:10 | YML 파일 | 10KB |
| opencv_videoio_ffmpeg430.dll | 2020-07-14 오전 9:52 | 응용 프로그램 확장 | 20,714KB |
| README | 2021-09-04 오후 6:15 | 텍스트 문서 | 2KB |

(DarkLabel Launch Screen)




**First, I add the classes I will write through the darklabel.yml file before annobrating the video.**

| | | | |
|---|---|---|---|
| DarkLabel | 2021-09-04 오후 6:12 | 응용 프로그램 | 20,261KB |
| darklabel | 2024-11-21 오전 4:16 | SAV 파일 | 1KB |
| darklabel | 2024-11-21 오전 3:10 | YML 파일 | 10KB |
| opencv_videoio_ffmpeg430.dll | 2020-07-14 오전 9:52 | 응용 프로그램 확장 | 20,714KB |
| README | 2021-09-04 오후 6:15 | 텍스트 문서 | 2KB |

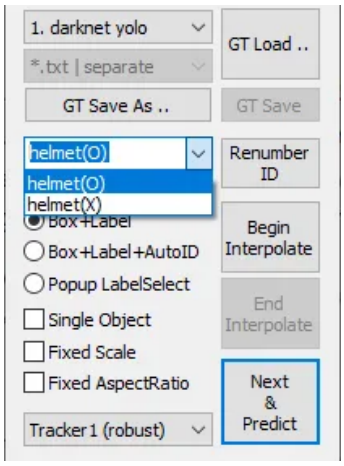
**Make my_classes2 in the yaml file and add helmet(O) and helmet(X) as class names that can distinguish between wearing and not wearing helmets.**

```
%YAML:1.0
---
## Default Settings
media_path_root: "H:₩₩darklabel_test₩₩media"          # if specified, image/video files are opened in this folder by default
gt_path_root: "H:₩₩darklabel_test₩₩gt"          # if specified, gt files are loaded and saved in $gt_path_root₩ by default
auto_gt_load: 0                # if true, gt is loaded from $gt_path_root₩ when media is opened (you have to select gt format first be
gt_file_ext: "xml"             # default gt save file format (supported formats: xml, txt, csv)
gt_merged: 0                   # 0: save gt as separate file for each image, 1: save gt in one file
delimiter: ","                 # separating delimiter of gt datum (it's effective only when gt is saved in txt)
database_name: "Unknown"       # it is used when saving gt data in xml format: <database>database_name</database>
classes_set: "autonomous_driving_classes"   # predefined classes set (tag name of classes set)

## Predefined Classes Sets (you can define and add your own classes set here)
my_classes1: ["lane", "vehicle", "bicycle", "motorbike", "animal", "tree", "building"]
my_classes2: ["helmet(O)","helmet(X)"]
autonomous_driving_classes: ["lane"]
coco_classes: ["person", "bicycle", "car", "motorcycle", "airplane", "bus", "train", "truck", "boat", "traffic light", "fire hydrant", "stop sign'
voc_classes: ["aeroplane", "bicycle", "bird", "boat", "bottle", "bus", "car", "cat", "chair", "cow", "diningtable", "dog", "horse", "motorbike'
```

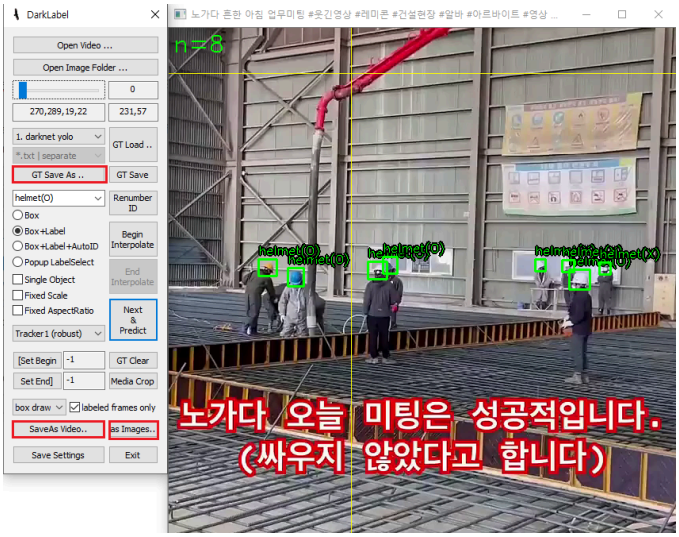
**Now, add "my_classes2" to see the class set in the DarkLabel GUI when announcing, and set the name to "darknet yolo"**

```
format0:    # pascal voc & imagenet (predefined format)
  fixed_filetype: 1              # if specified as true, save setting isn't changeable in GUI
  data_fmt: [cname, difficult, x1, y1, x2, y2]
  gt_file_ext: "xml"             # if not specified, default setting is used
  gt_merged: 0                   # if not specified, default setting is used
  classes_set: "voc_classes"     # if not specified, default setting is used
  name: "pascal voc"             # if not specified, "[fmt%d] $data_fmt" is used as default format name

format1:    # darknet yolo (predefined format)
  fixed_filetype: 1              # if specified as true, save setting isn't changeable in GUI
  data_fmt: [classid, ncx, ncy, nw, nh]
  gt_file_ext: "txt"             # if not specified, default setting is used
  gt_merged: 0                   # if not specified, default setting is used
  delimiter: " "                 # if not spedified, default delimiter(',') is used
  classes_set: "my_classes2"     # if not specified, default setting is used
  name: "darknet yolo"           # if not specified, "[fmt%d] $data_fmt" is used as default format name
```

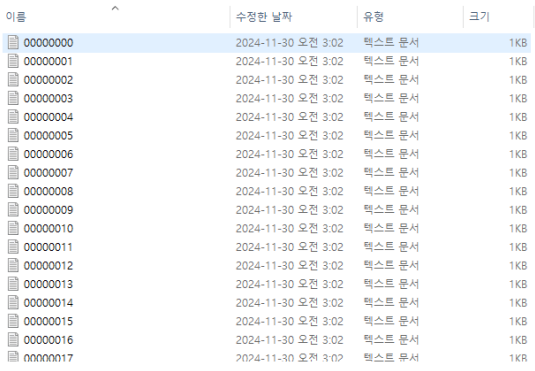
**It can be seen that two classes set separately from the classes "darknet yolo" have been added.**

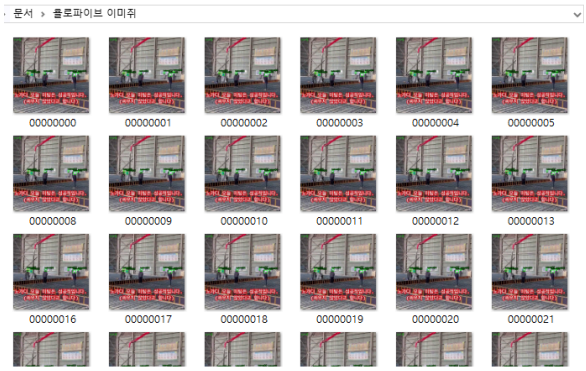



**After finishing labeling in DarkLabel, press the "as Images" button to select an image folder to save the converted image. Then, press the "GT Save As" button to create a separate label storage folder and store it in the folde**




**Now you can see that there are separate labels and converted txt files and image files in the image folder.**

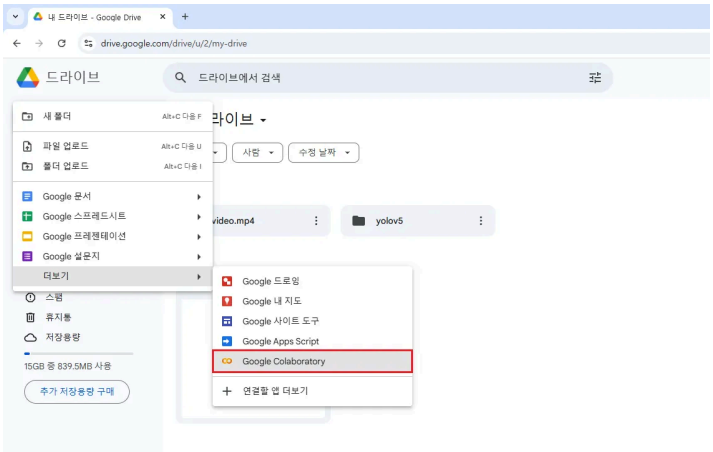



---

# Nvidia Jetson Nano Training Process

**Enter Google Drive and Install Google Collaboration for Image Learning**



**Run Colab and enter the code that connects to the Google drive in the command window prompt.**

```
# prompt: 구글 드라이브랑 연결

from google.colab import drive
drive.flush_and_unmount()  # 기존 마운트 해제
drive.mount('/content/drive')  # 다시 마운트

%cd /content/drive/MyDrive

%pwd
```

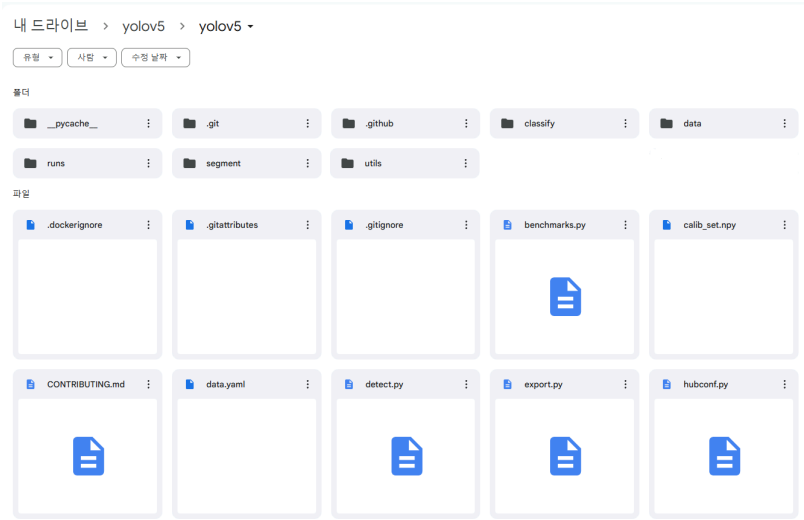
## Yolov5

**Access yolov5 on Google Drive, download it, and install all the necessary libraries in bulk using the Requirements.txt file.**

```
# 기존에 설치를 완료한 경우에는 해당 경로로 이동만 하면 됩니다.
%cd /content/drive/MyDrive/yolov5

##
#clone YOLOv5 and
!git clone https://github.com/ultralytics/yolov5  # clone repo
%cd yolov5
%pip install -qr requirements.txt # install dependencies

!pip install Pillow==10.3
```

**After run the code "!pip install Pillow==10.3" and run again that connects to the Google drive in the command window prompt**

**A folder called yolov5 is created and contains gitclone files in it.**

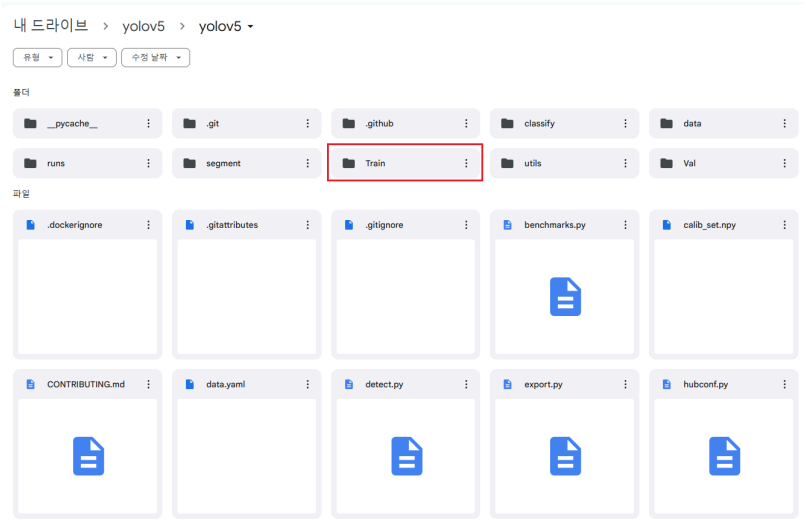


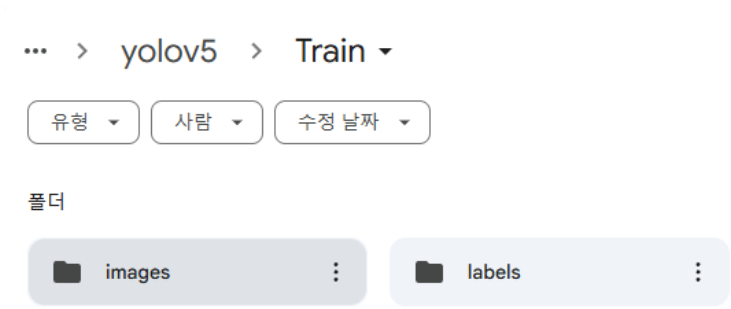

## Managing Image and Label Files

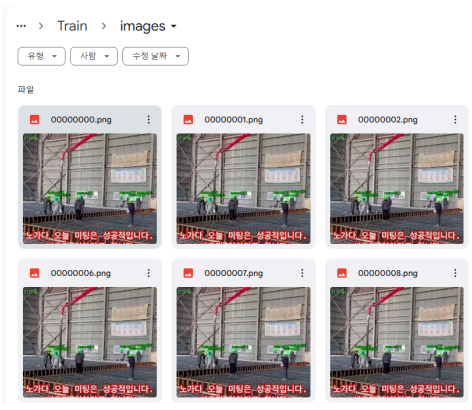
**Enter the folder creation code for managing image and label files.**

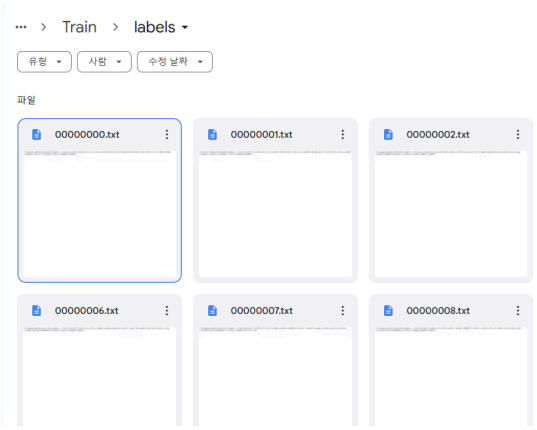
```
!mkdir -p Train/labels
!mkdir -p Train/images
!mkdir -p Val/labels
!mkdir -p Val/images
```

**Train and Val folders that may manage image and label files are created in the previously created yolov5 folder, and the converted image and label generated by Darklabel are uploaded to the image and label folder of the Train folder.**

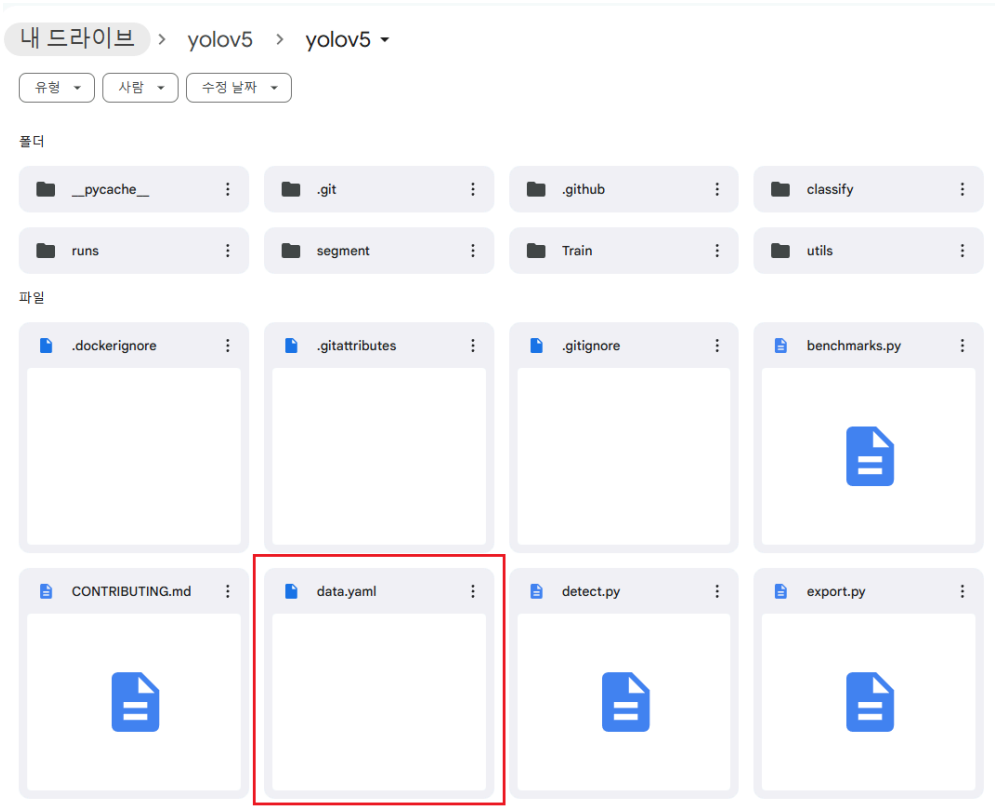


**After that, put the yolov5n.pt file that came with the guide in the yolov5 folder and the data.yaml file that you modified to match the name of the class you labeled.**




```
##검증 데이터 만들기
import os
import shutil
from sklearn.model_selection import train_test_split

def create_validation_set(train_path, val_path, split_ratio=0.3):
    """
    Train 데이터의 일부를 Val로 이동
    """
    # 필요한 디렉토리 생성
    os.makedirs(os.path.join(val_path, 'images'), exist_ok=True)
    os.makedirs(os.path.join(val_path, 'labels'), exist_ok=True)

    # Train 이미지 리스트 가져오기
    train_images = os.listdir(os.path.join(train_path, 'images'))
    train_images = [f for f in train_images if f.endswith(('.jpg', '.jpeg', '.png'))]

    # Train/Val 분할
    _, val_images = train_test_split(train_images,
                                     test_size=split_ratio,
                                     random_state=42)

    # Val로 파일 복사
    for image_file in val_images:
        # 이미지 복사
        src_image = os.path.join(train_path, 'images', image_file)
```

```
        dst_image = os.path.join(val_path, 'images', image_file)
        shutil.copy2(src_image, dst_image)

        # 라벨 파일 복사
        label_file = os.path.splitext(image_file)[0] + '.txt'
        src_label = os.path.join(train_path, 'labels', label_file)
        dst_label = os.path.join(val_path, 'labels', label_file)
        if os.path.exists(src_label):
            shutil.copy2(src_label, dst_label)

    print(f"Created validation set with {len(val_images)} images")

# 실행
train_path = '/content/drive/MyDrive/yolov5/yolov5/Train'
val_path = '/content/drive/MyDrive/yolov5/yolov5/Val'

create_validation_set(train_path, val_path)
```

```
def check_dataset():
    train_path = '/content/drive/MyDrive/yolov5/yolov5/Train'
    val_path = '/content/drive/MyDrive/yolov5/yolov5/Val'

    # Train 데이터 확인
    train_images = len(os.listdir(os.path.join(train_path, 'images')))
    train_labels = len(os.listdir(os.path.join(train_path, 'labels')))

    # Val 데이터 확인
    val_images = len(os.listdir(os.path.join(val_path, 'images')))
    val_labels = len(os.listdir(os.path.join(val_path, 'labels')))

    print("Dataset status:")
    print(f"Train - Images: {train_images}, Labels: {train_labels}")
    print(f"Val - Images: {val_images}, Labels: {val_labels}")

# 데이터셋 상태 확인
check_dataset()
```

## Start of yolov5 model training

**Enter the code to import the required library to start learning the model.**

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

```
%pwd
```

```
import numpy as np
import tensorflow as tf
import os
from PIL import Image
```

```
from tensorflow.python.eager.context import eager_mode

def _preproc(image, output_height=512, output_width=512, resize_side=512):
    ''' imagenet-standard: aspect-preserving resize to 256px smaller-side, then central-crop to
    with eager_mode():
        h, w = image.shape[0], image.shape[1]
        scale = tf.cond(tf.less(h, w), lambda: resize_side / h, lambda: resize_side / w)
        resized_image = tf.compat.v1.image.resize_bilinear(tf.expand_dims(image, 0), [int(h*scal
        cropped_image = tf.compat.v1.image.resize_with_crop_or_pad(resized_image, output_height,
        return tf.squeeze(cropped_image)

def Create_npy(imagespath, imgsize, ext) :
    images_list = [img_name for img_name in os.listdir(imagespath) if
                os.path.splitext(img_name)[1].lower() == '.'+ext.lower()]
    calib_dataset = np.zeros((len(images_list), imgsize, imgsize, 3), dtype=np.float32)

    for idx, img_name in enumerate(sorted(images_list)):
        img_path = os.path.join(imagespath, img_name)
        try:
            # 파일 크기가 정상적인지 확인
            if os.path.getsize(img_path) == 0:
                print(f"Error: {img_path} is empty.")
                continue

            img = Image.open(img_path)
            img = img.convert("RGB")  # RGBA 이미지 등 다른 형식이 있을 경우 강제로 RGB로 변환
            img_np = np.array(img)

            img_preproc = _preproc(img_np, imgsize, imgsize, imgsize)
            calib_dataset[idx,:,:,:] = img_preproc.numpy().astype(np.uint8)
            print(f"Processed image {img_path}")

        except Exception as e:
            print(f"Error processing image {img_path}: {e}")

    np.save('calib_set.npy', calib_dataset)
```

```
# "cannot identify image file" 에러가 발생하는 경우, PILLOW Version을 "!pip install Pillow==10.1"
Create_npy('/content/drive/MyDrive/yolov5/yolov5/Train/images', 512, 'jpg')
```

**After inputting the above codes sequentially, inputting an instruction code to learn the labeled photos, and then waiting.**

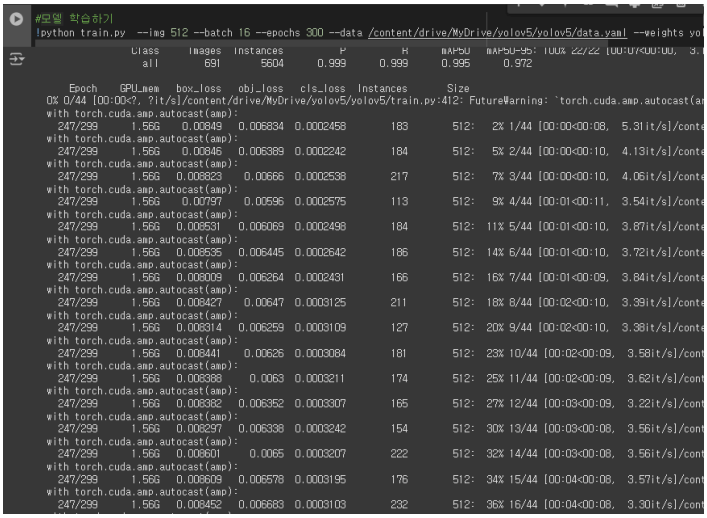
```
#모델 학습하기
!python train.py  --img 640 --batch 16 --epochs 300 --data /content/drive/MyDrive/yolov5/yolov5/
```

> **img 640 : Set the input image size (640x640)**

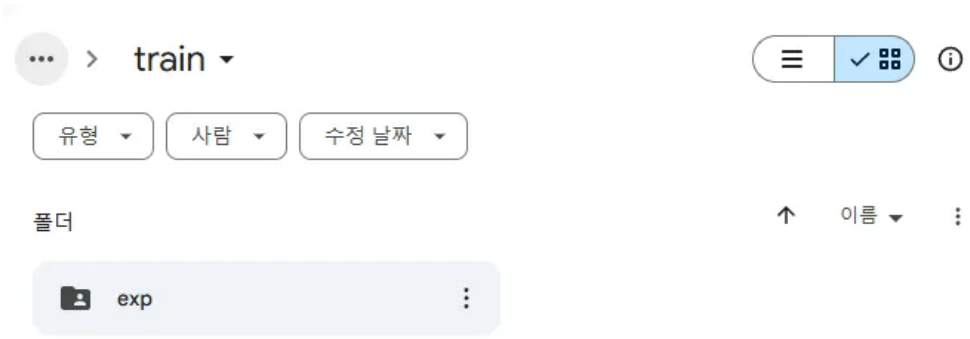
> **batch 16 : batch size setting (number of images processed at a time)**

> **epochs 2000: Set the total number of epochs to learn**

> **data /content/drive/MyDrive/yolov5/yolov5/data.yaml: Specify the path to the yaml file that contains settings for the dataset and model configuration (path to the data.yaml file that you uploaded earlier)**

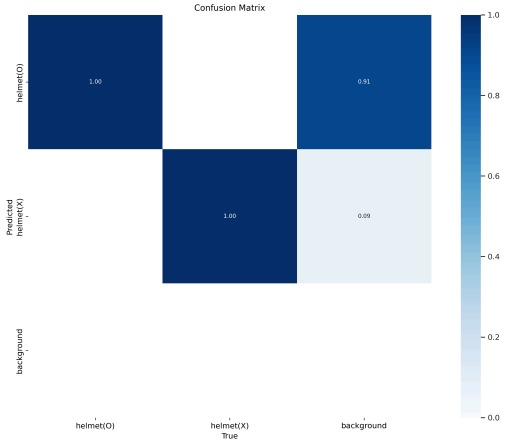


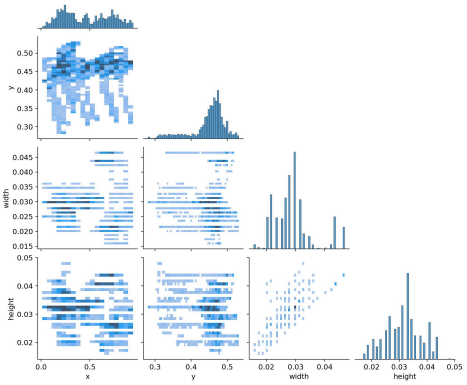

**After the learning is completed, it can be confirmed that the learning has been performed in the runs → train → exp folder in the yolov5 folder.**

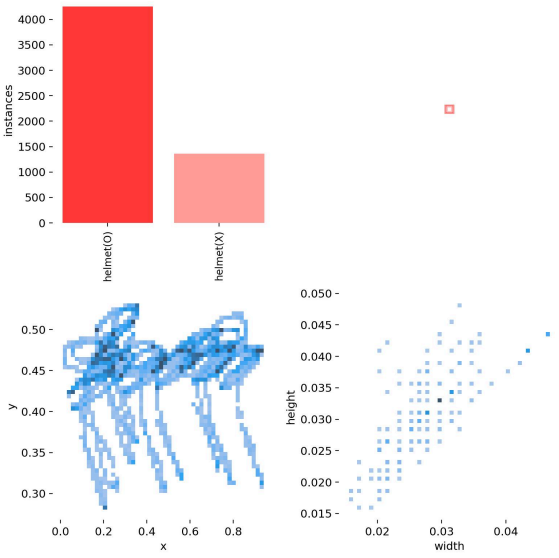



**The learning outcomes are as follows.**

https://drive.google.com/drive/folders/1Pnznlpp3lVJ07zvkF9WLPl-XC5J3x-aC?usp=drive_link

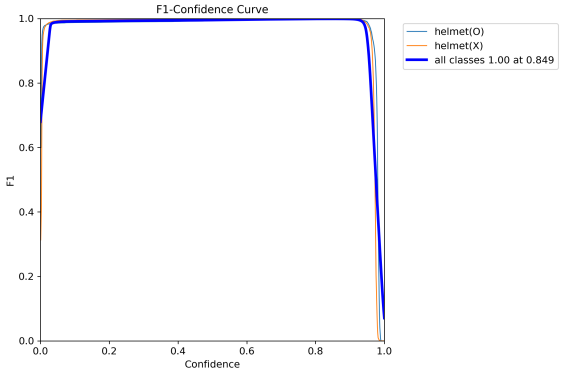


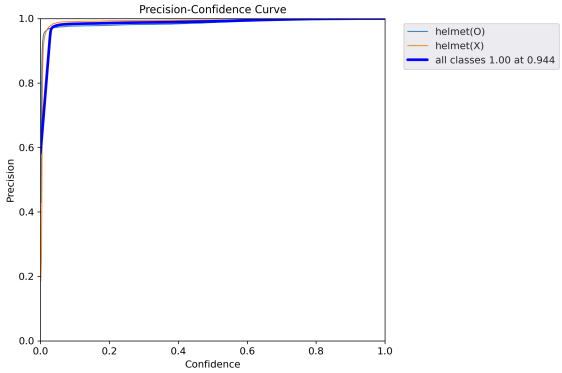




## F1 Curve

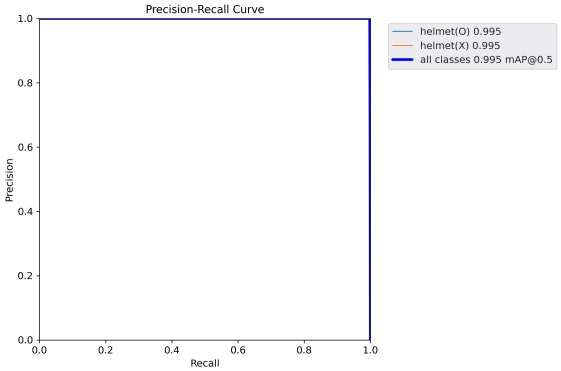
## P Curve

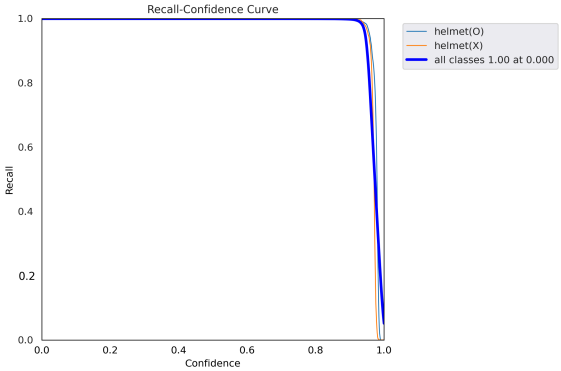


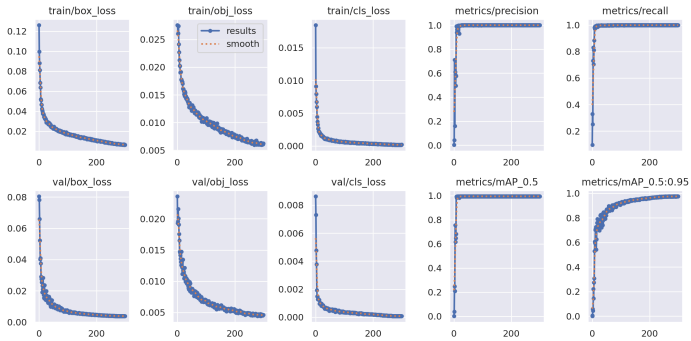

## PR Curve

## R Curve

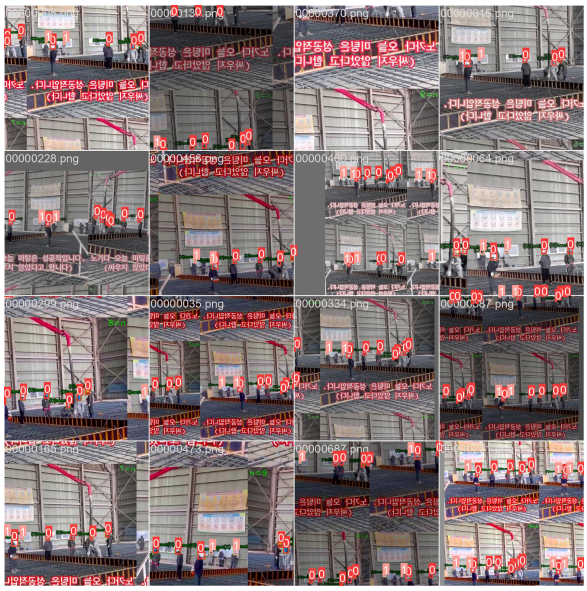


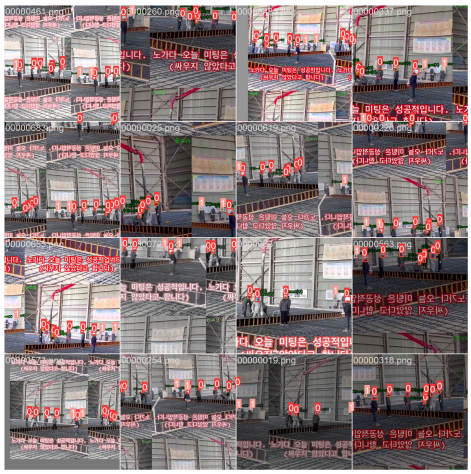




## Results




## Train batch





**Since it is necessary to check the learning results after learning, there is a weights file in the exp file, of which best.pt is used to enter a command to check the learning results by going to detect.py**

```
!python detect.py --weights /content/drive/MyDrive/yolov5/yolov5/runs/train/exp/weights/best.pt
```

## Run video

**Enter a command to perform the learning result as an image**

```
!python detect.py --weights /content/drive/MyDrive/yolov5/yolov5/runs/train/exp/weights/best.pt
```

https://prod-files-secure.s3.us-west-2.amazonaws.com/361954c3-0e73-4464-90f4-d99314f50106/3b716a5e-f4ed-47ba-ab37-a9b6c71c103f/2024-12-02_13-49-08.mkv

https://drive.google.com/drive/folders/1cLVEpnCGNvk38q2oiVM-jbTBL_SsAtTS?usp=sharing