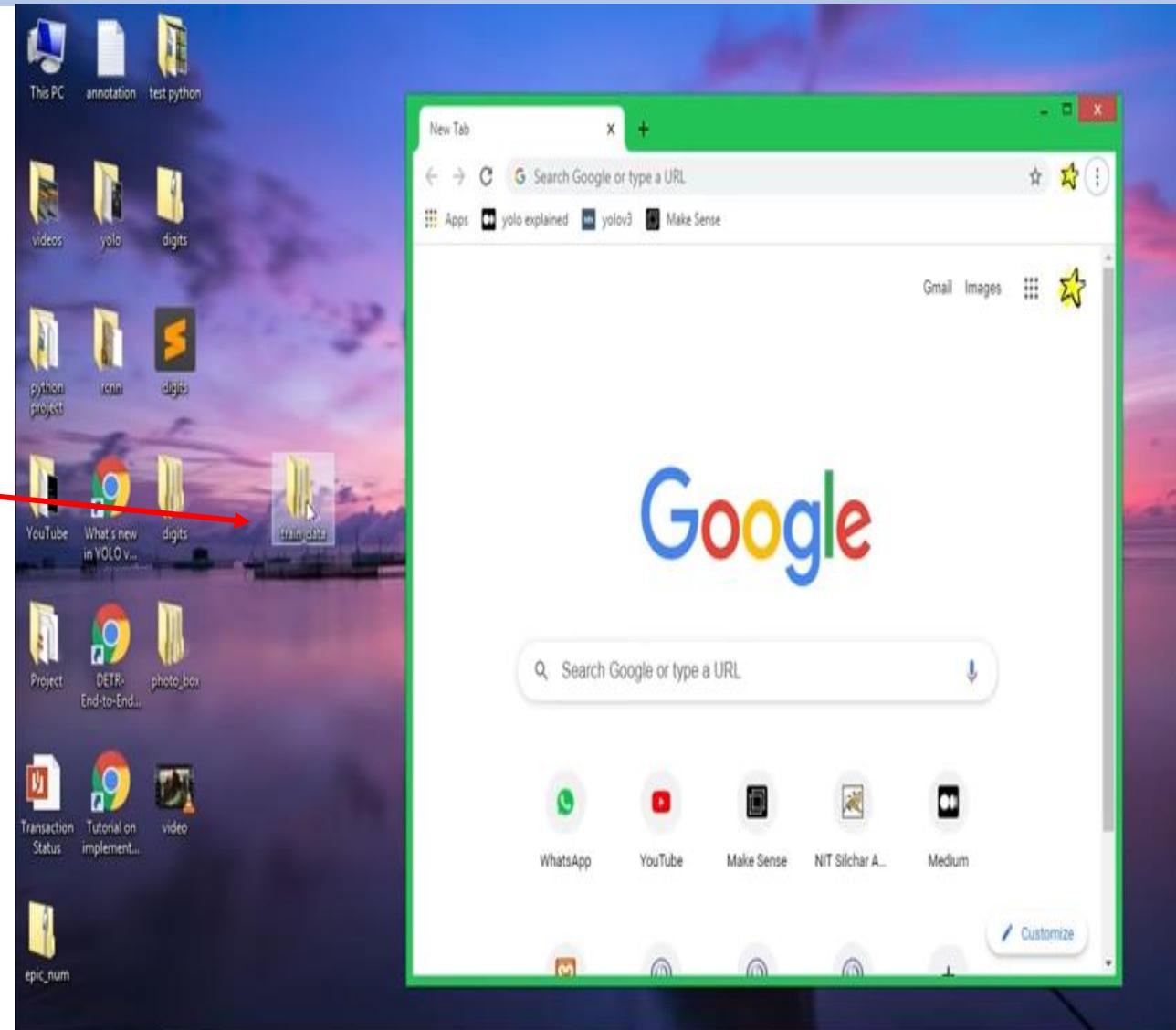


How to run YOLO

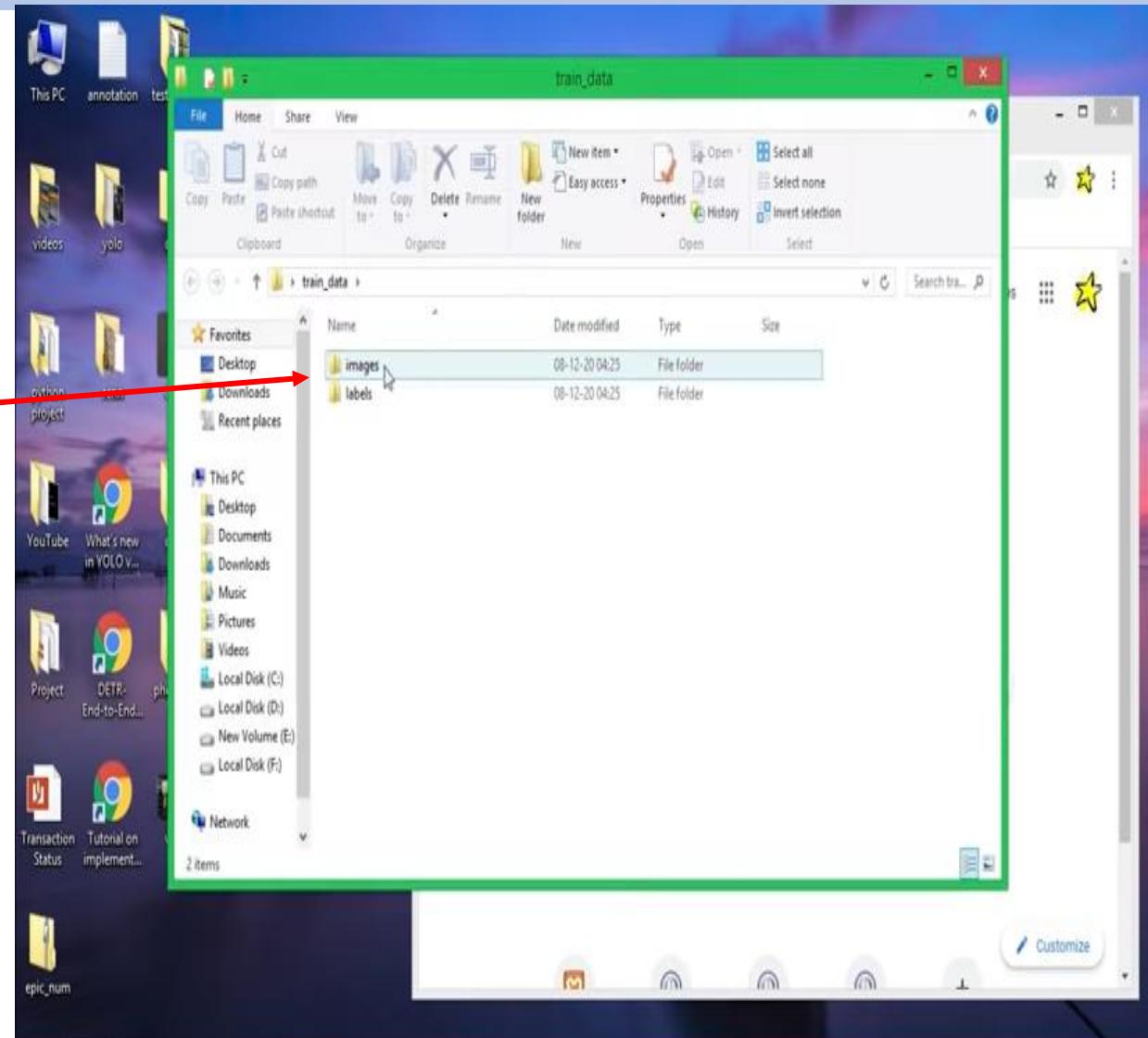
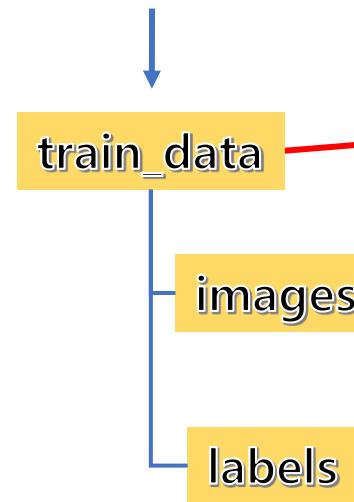
Data preparation : Make Your Own Dataset

Create Data Folder



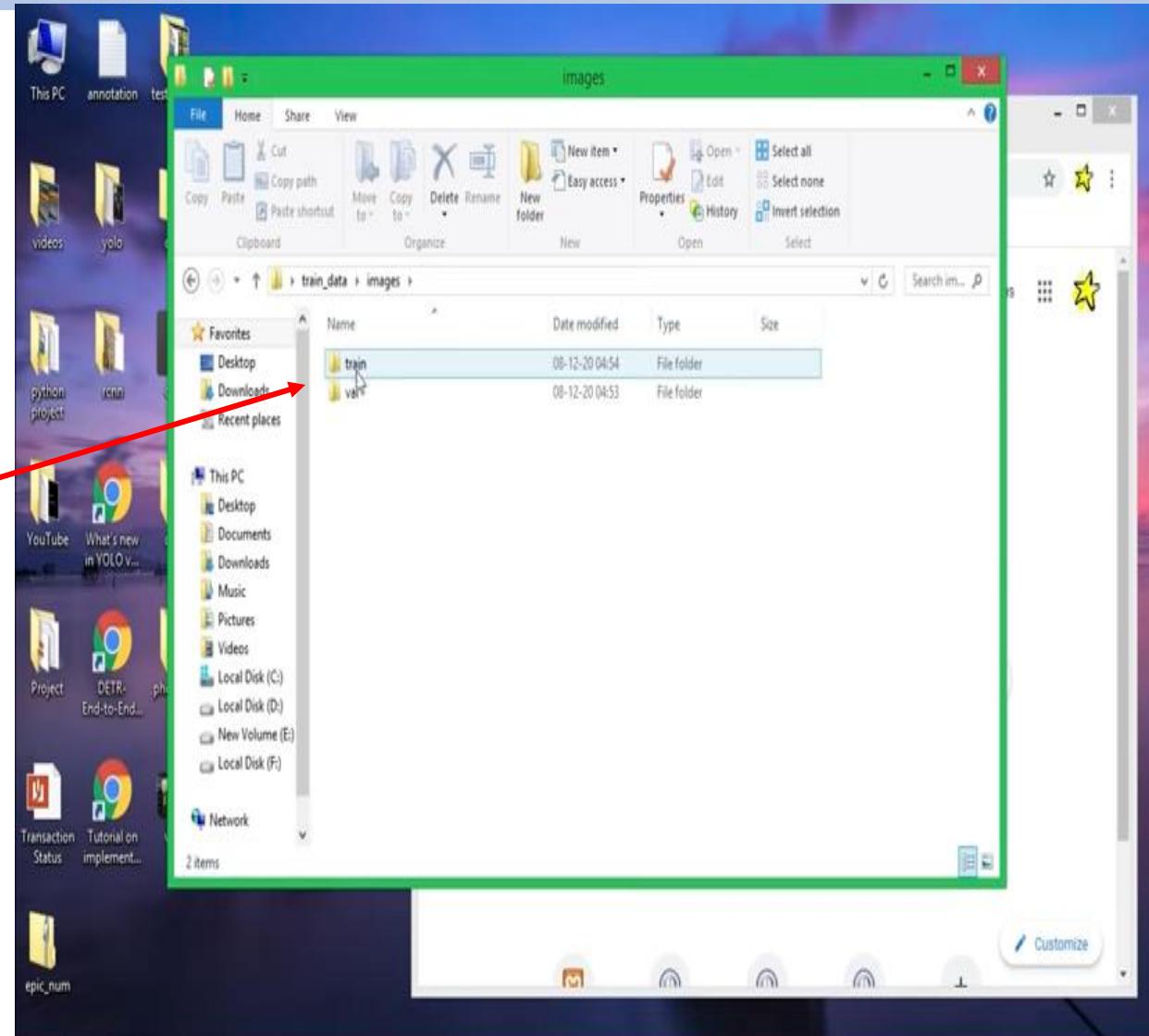
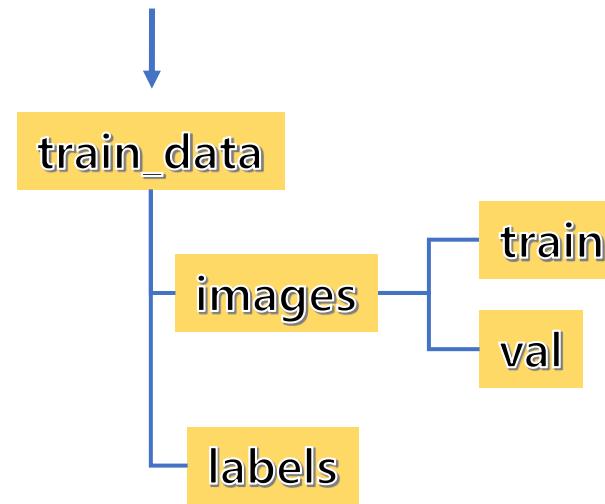
Data preparation : Make Your Own Dataset

Create Data Folder



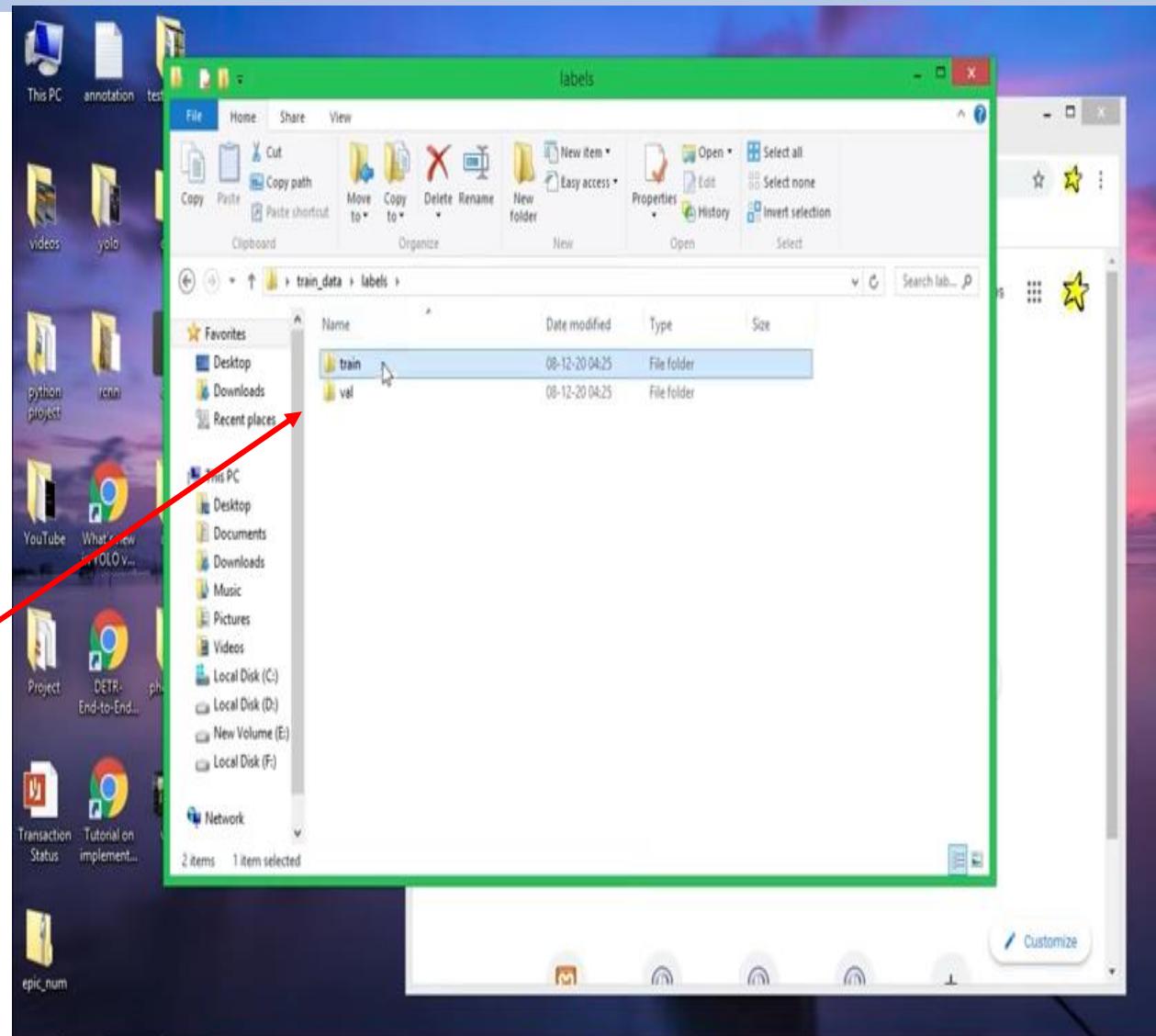
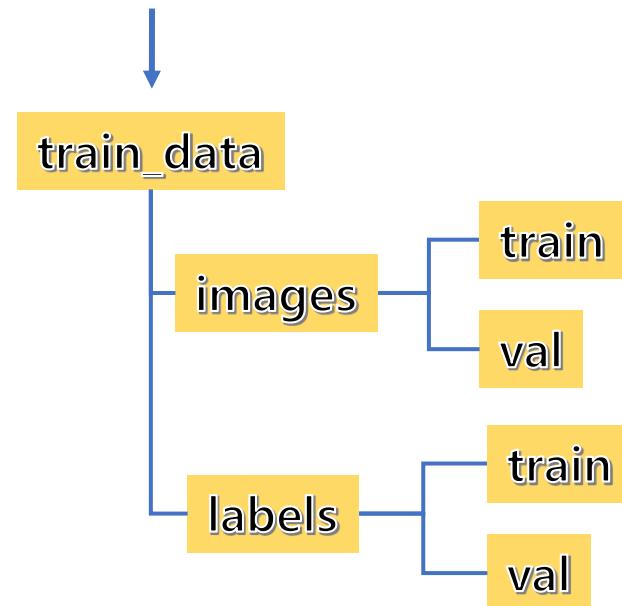
Data preparation : Make Your Own Dataset

Create Data Folder



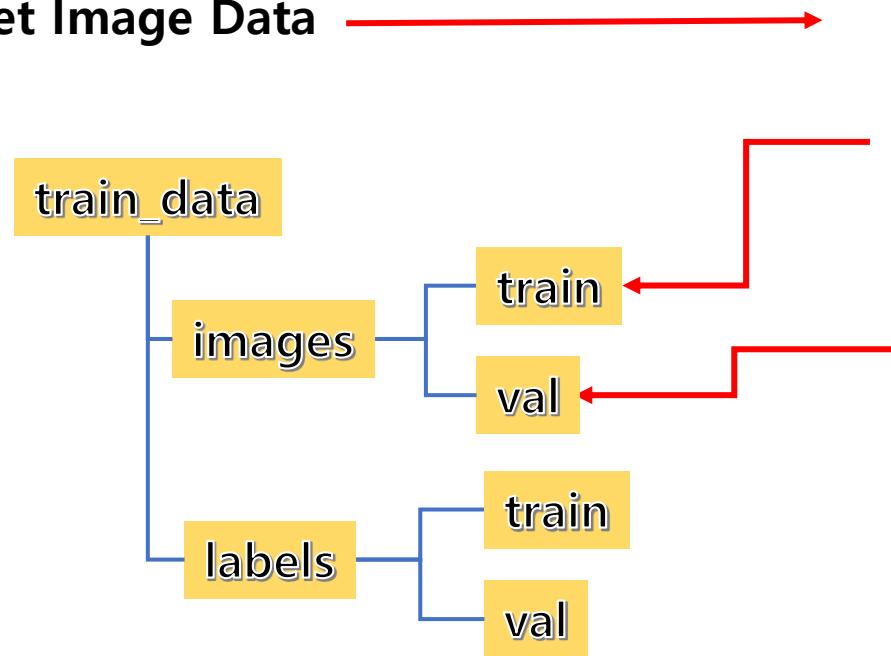
Data preparation : Make Your Own Dataset

Create Data Folder



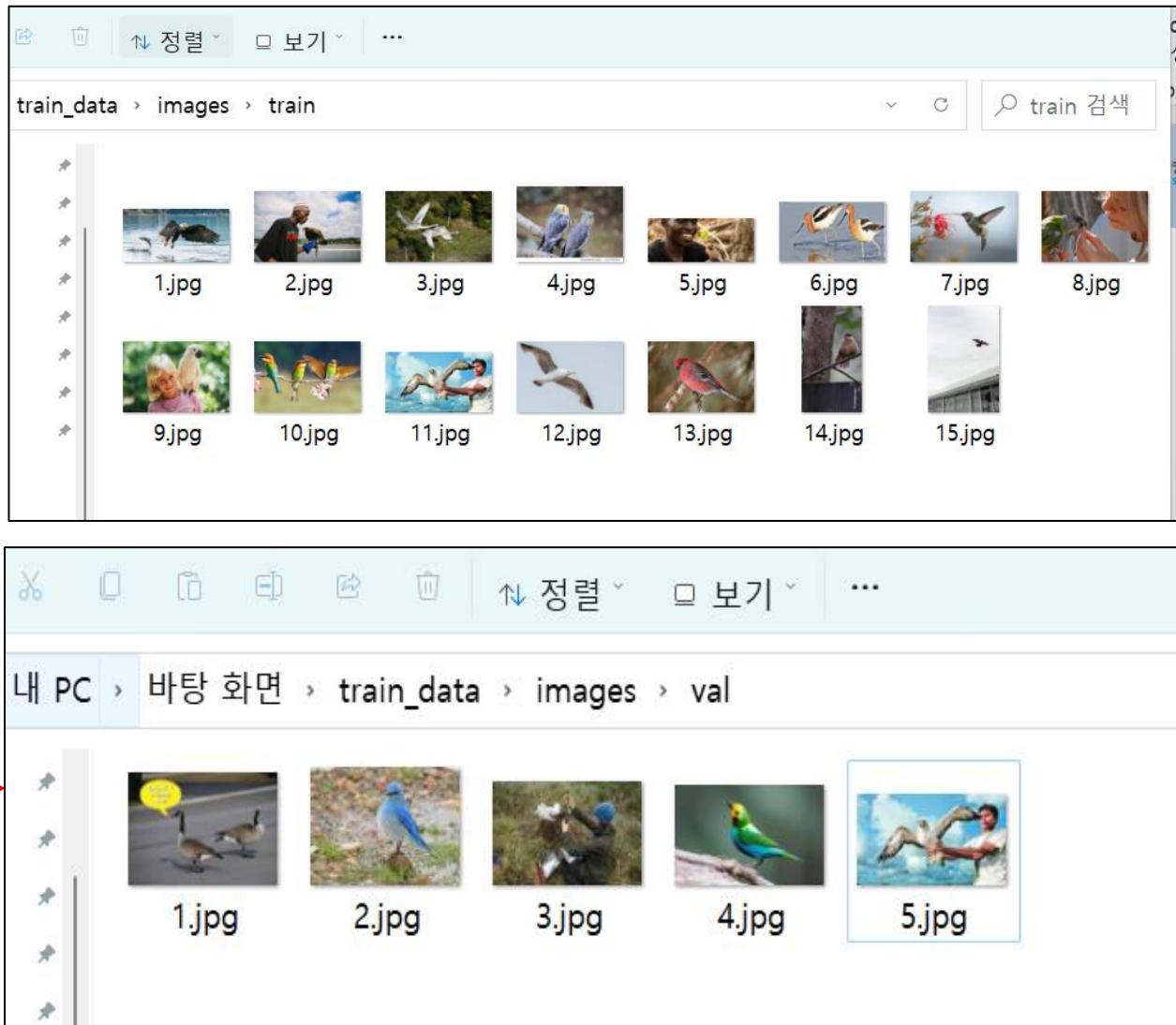
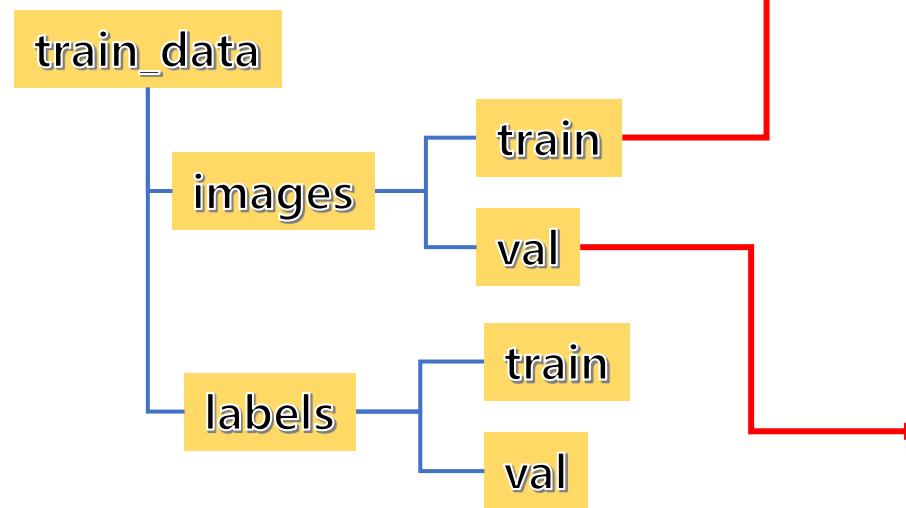
Data preparation : Make Your Own Dataset

Get Image Data



Data preparation : Make Your Own Dataset

Get Image Data

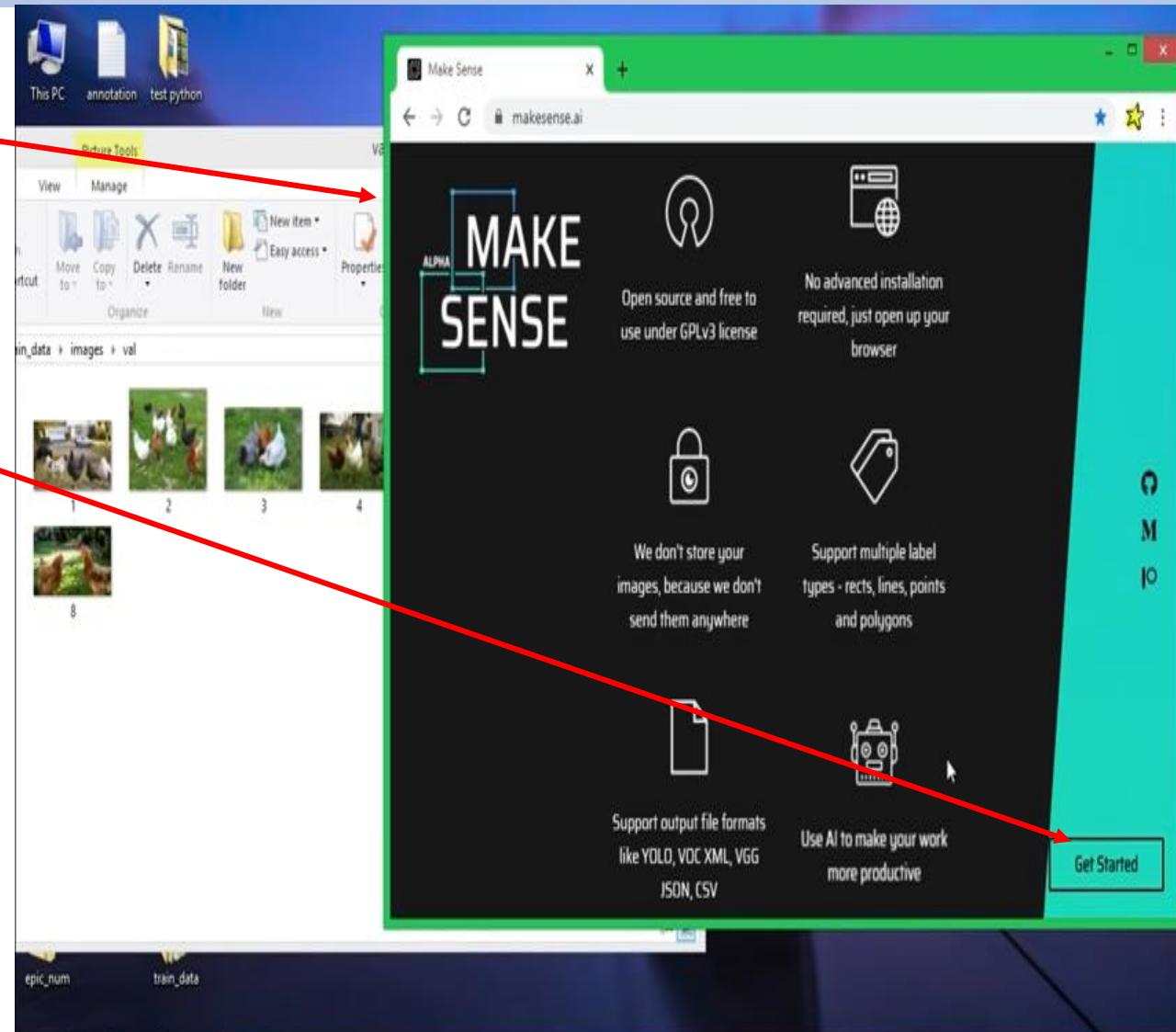


Data preparation : Make Your Own Dataset

Go to makesense.ai for data labeling

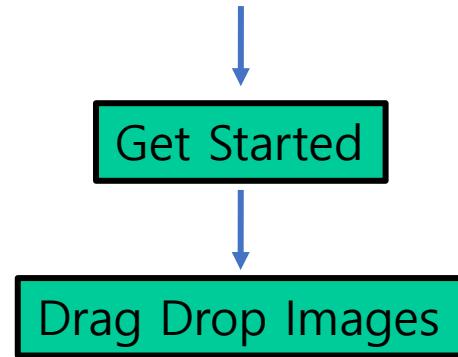
Get Started

<https://www.makesense.ai/>

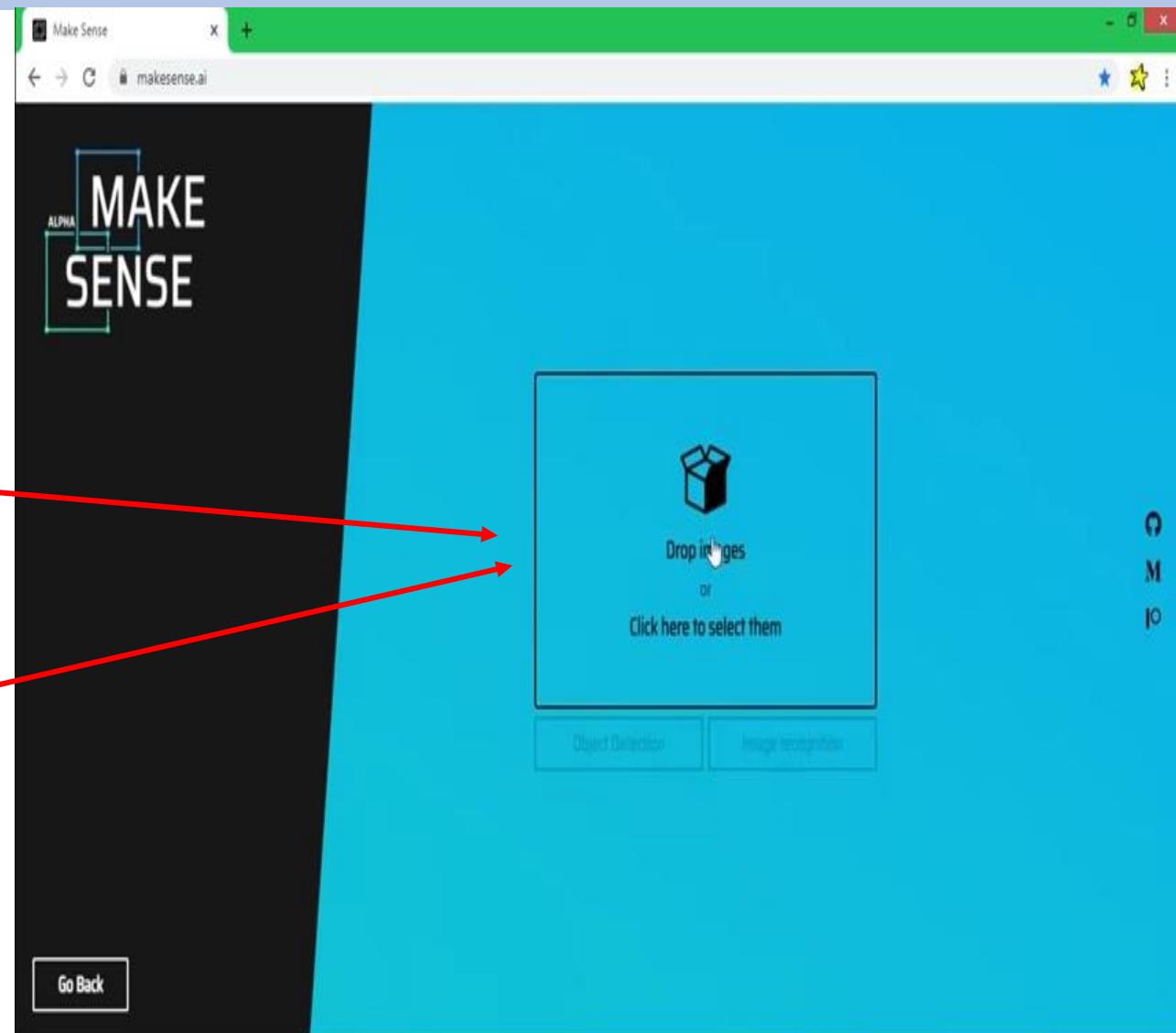
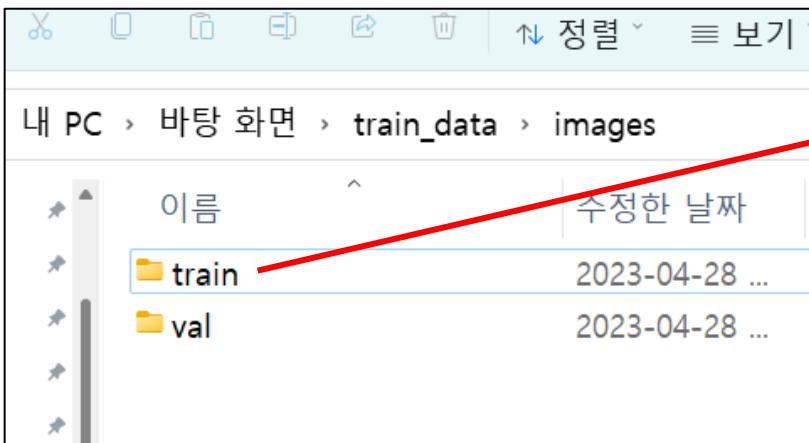


Data preparation : Make Your Own Dataset

Go to makesense.ai for data labeling

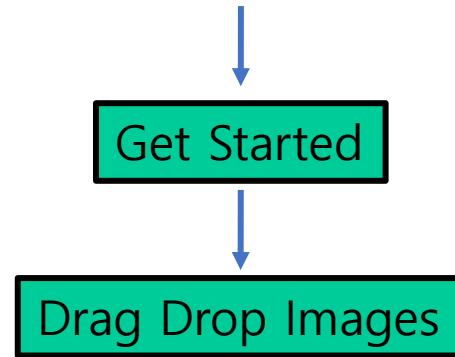


Drag train folder in images folder



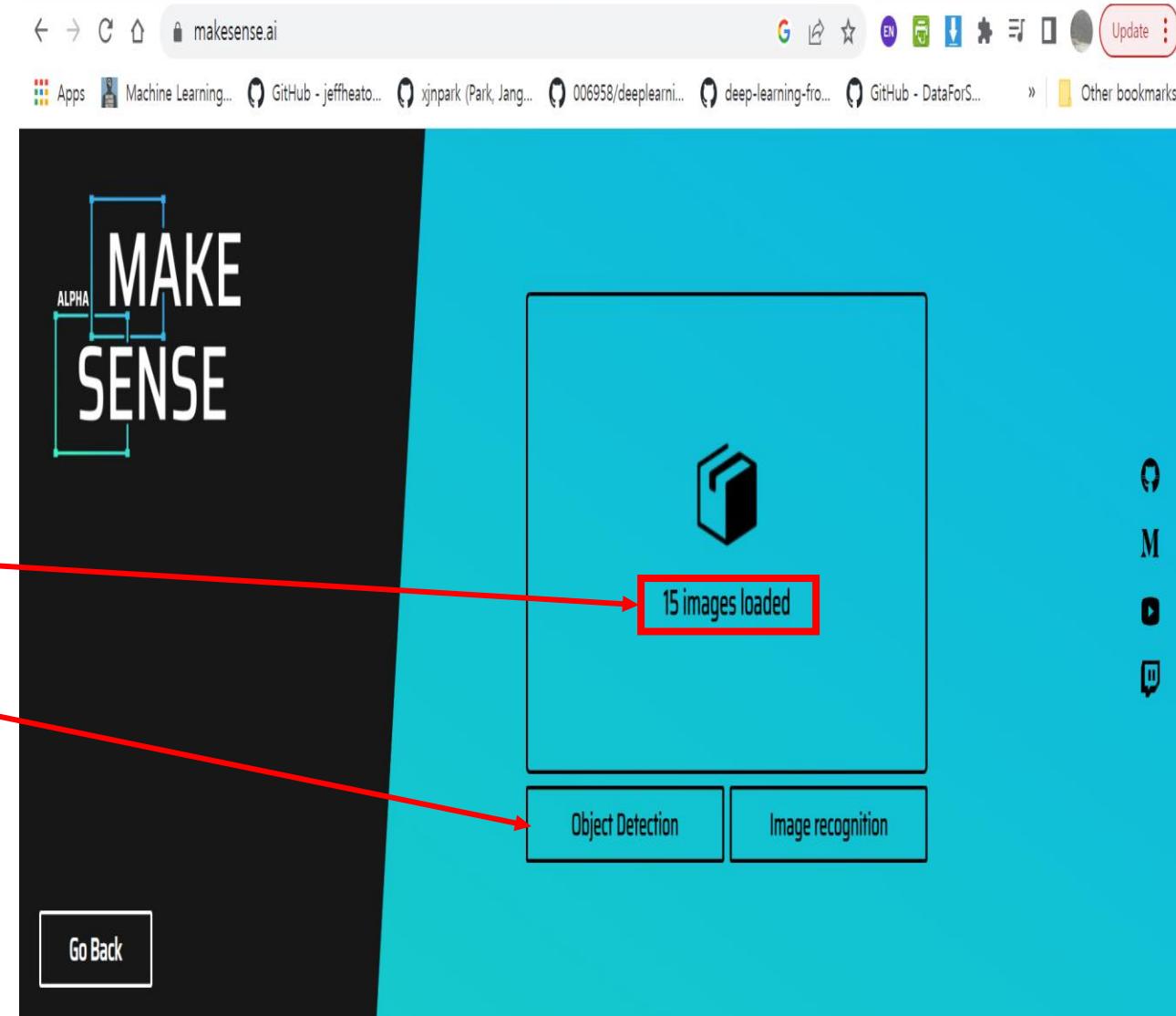
Data preparation : Make Your Own Dataset

Go to makesense.ai for data labeling



Drag train folder in images folder

Select : Object Detection



Data preparation : Make Your Own Dataset

Go to makesense.ai for data labeling



Get Started



Drag Drop Images

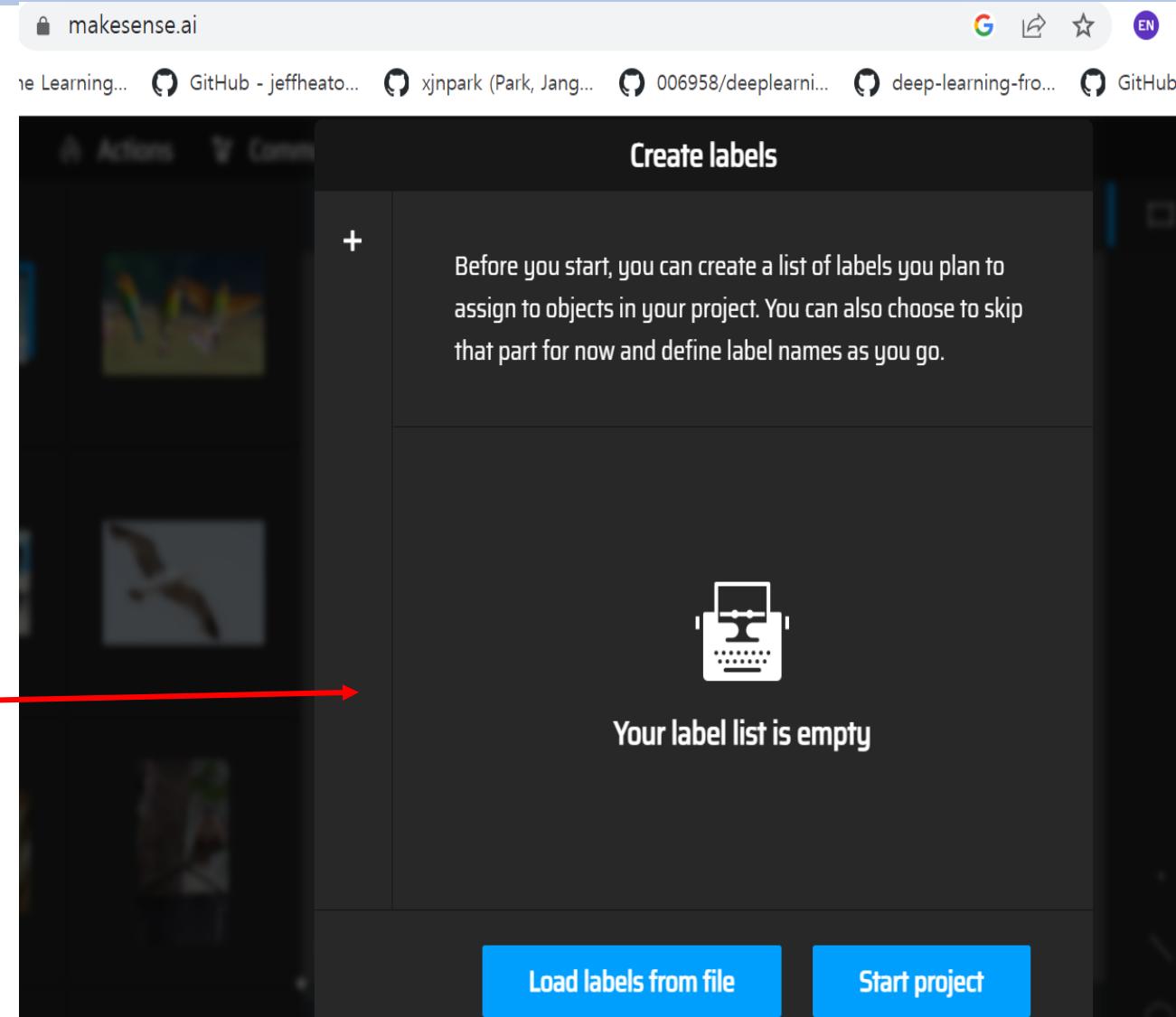
Drag train folder in images/train



Create Labels

Click + or enter, class1 name : bird

Click + or enter, class2 name : person



Data preparation : Make Your Own Dataset

Go to makesense.ai for data labeling



Get Started



Drag Drop Images

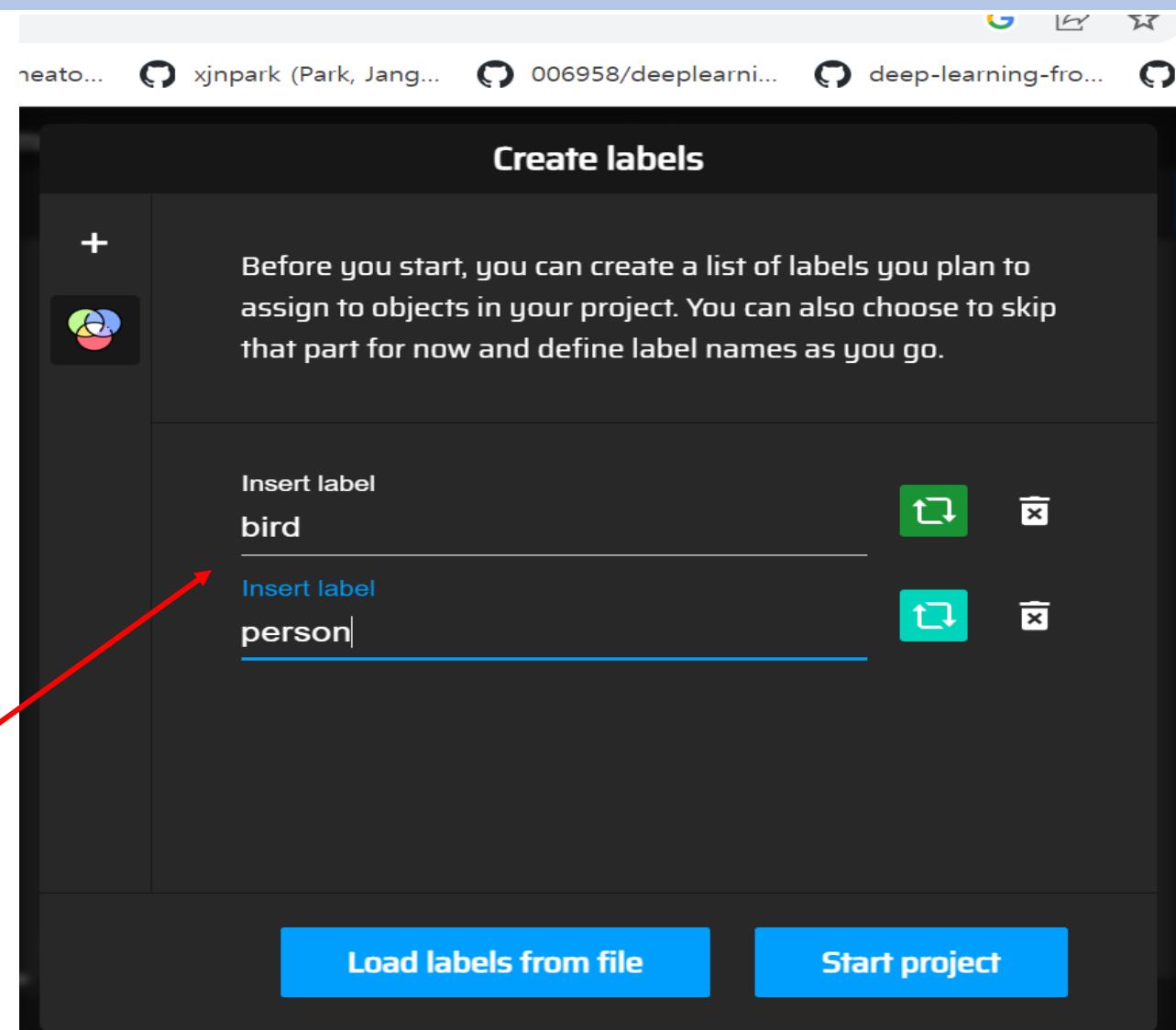
Drag train folder in images/train



Create Labels

Click + or enter, class1 name : bird

Click + or enter, class2 name : person



Data preparation : Make Your Own Dataset

Go to makesense.ai for data labeling



Get Started



Drag Drop Images

Drag train folder in images/train

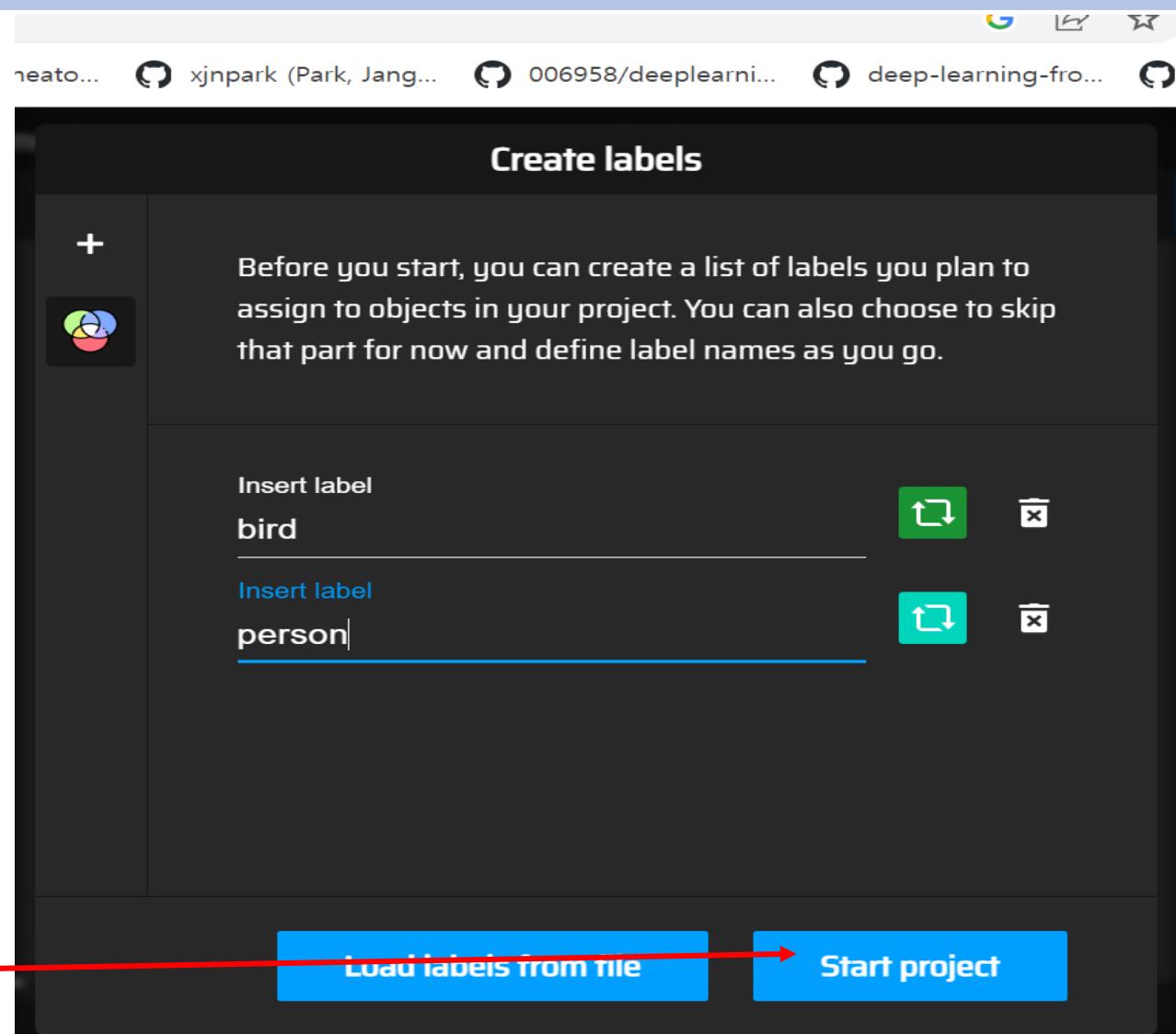


Create Labels

Click + or enter, class1 name : bird

Click + or enter, class2 name : person

Start project



Data preparation : Make Your Own Dataset

Apps Machine Learning... GitHub - jeffheato... xjnpark (Park, Jang... 006958/deeplearni... deep-learning-fro... GitHub - DataForS... » Other

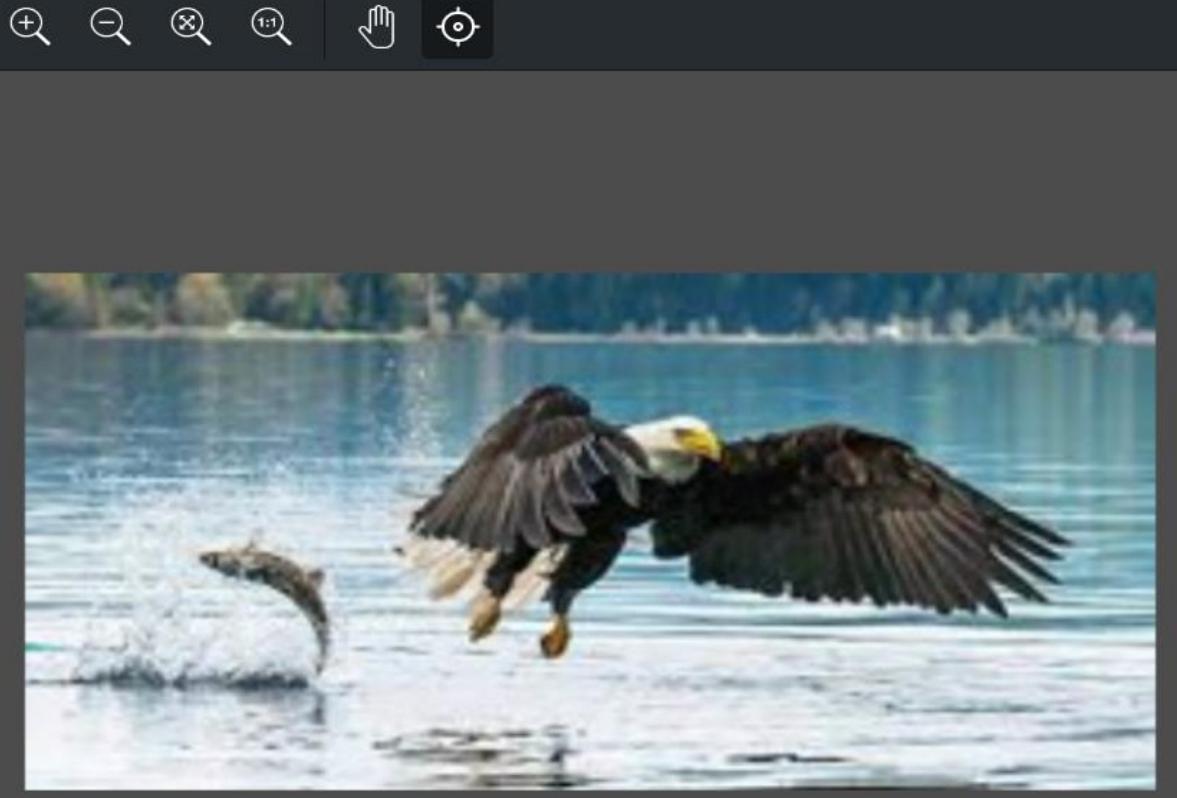
Make Sense

Actions

Community

Project Name: my-project-name

Images



Rect



draw your first
bounding box

Point

Line

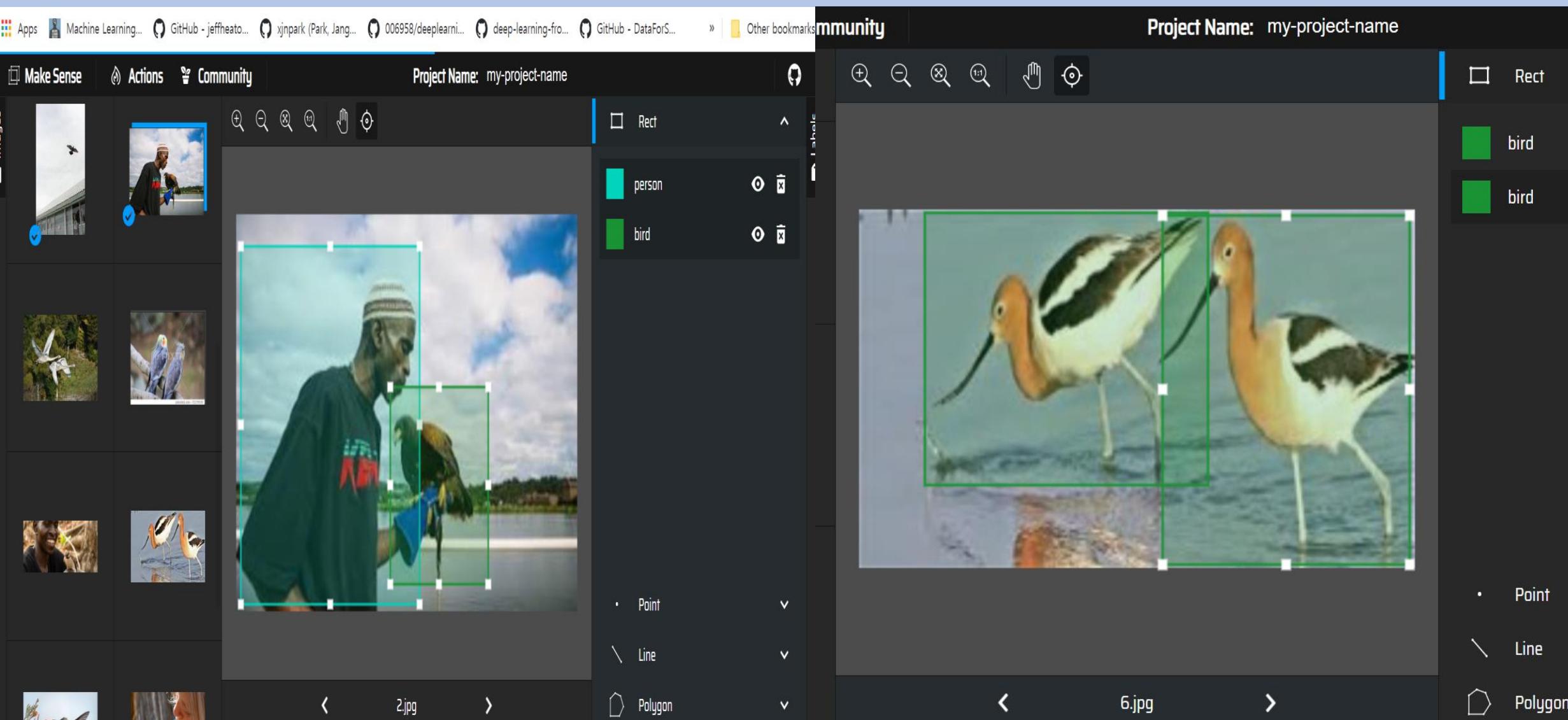
Polygon

<

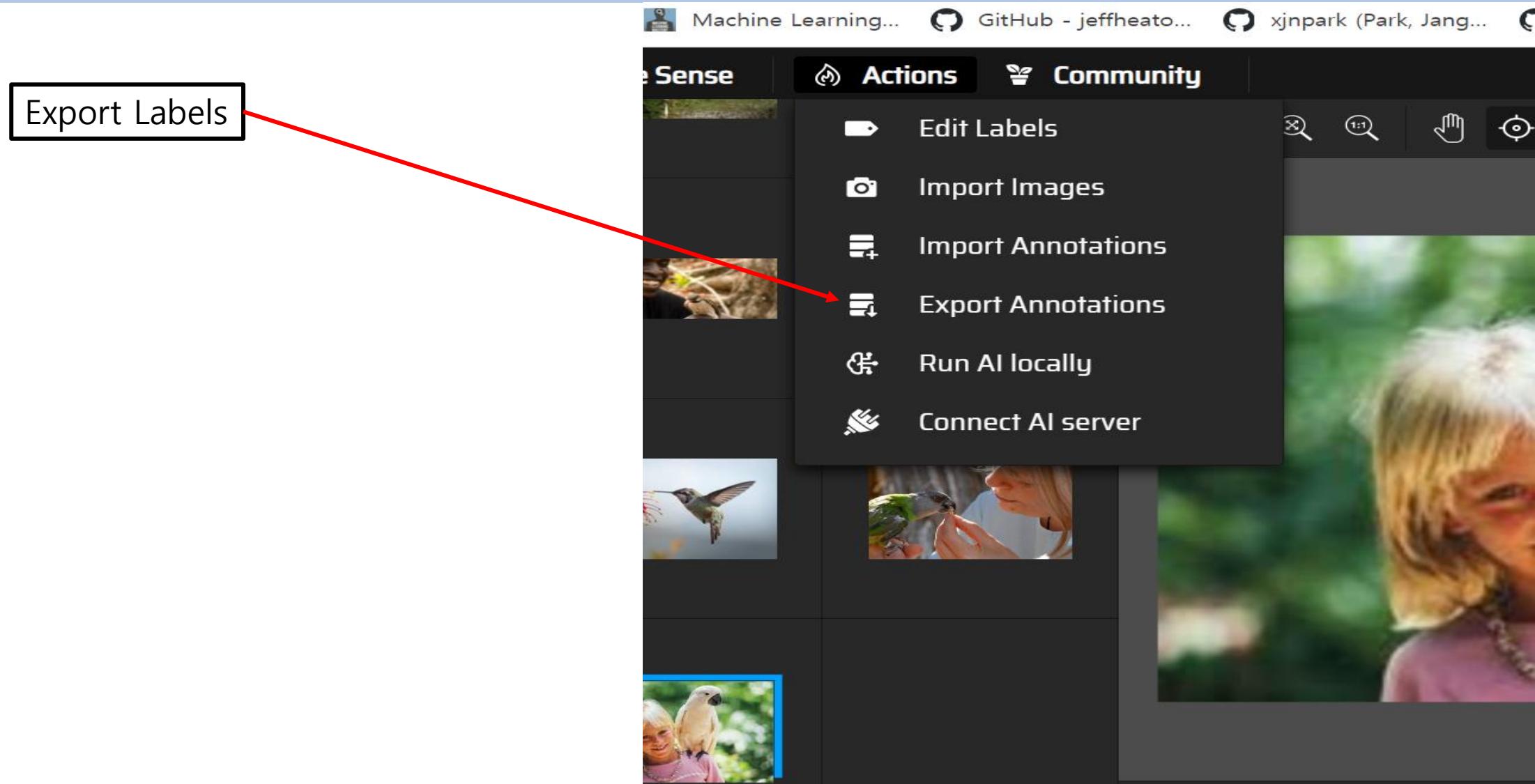
1.jpg

>

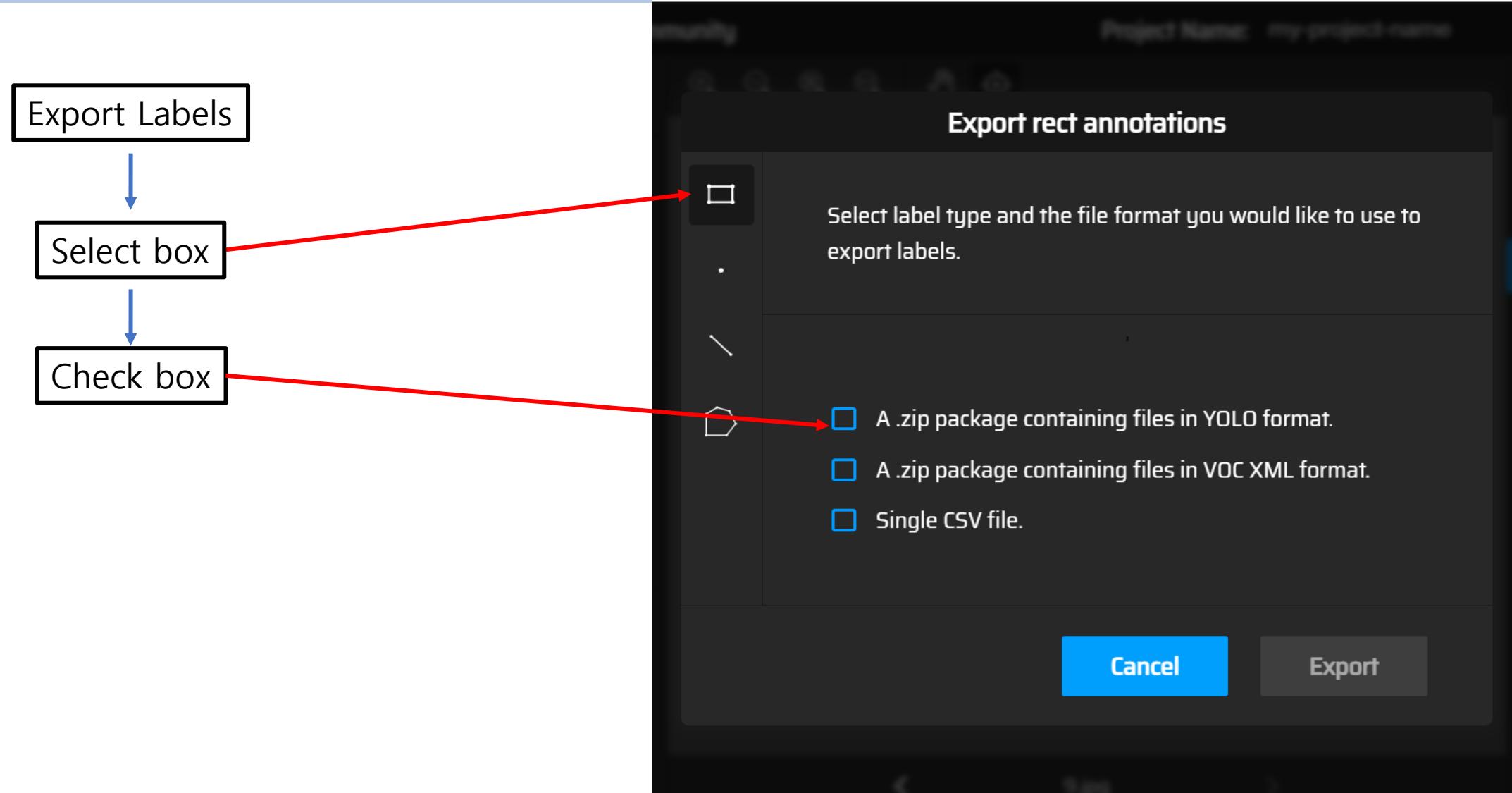
Data preparation : Make Your Own Dataset



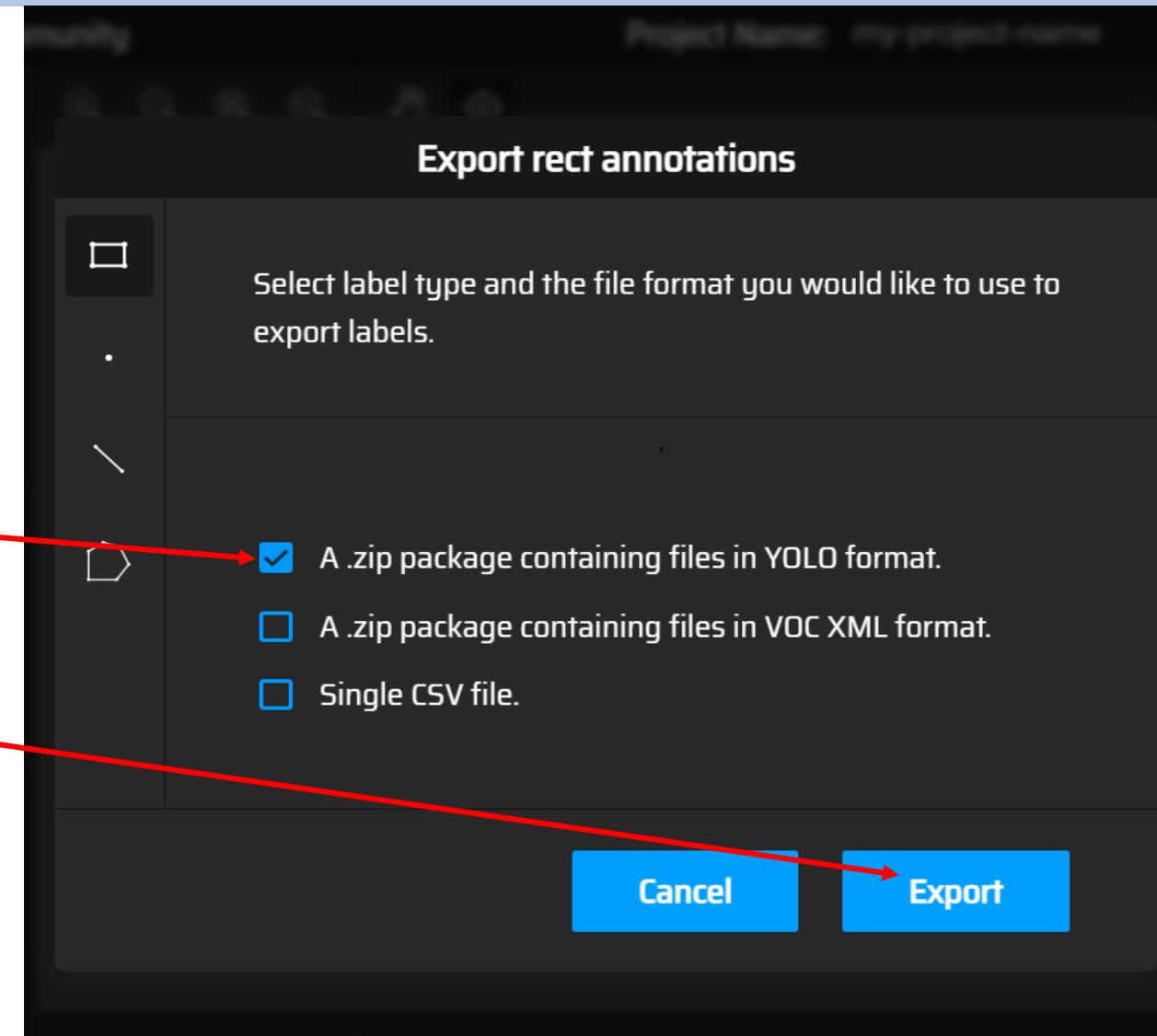
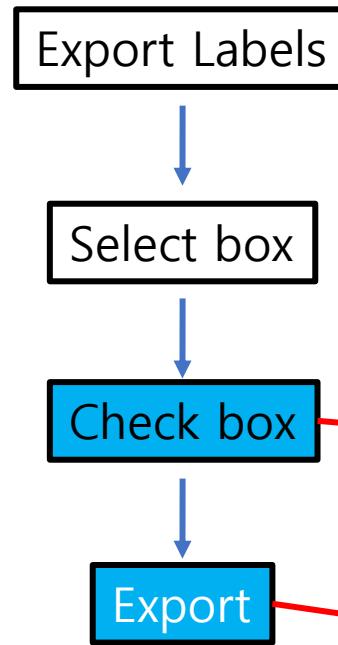
Data preparation : Make Your Own Dataset



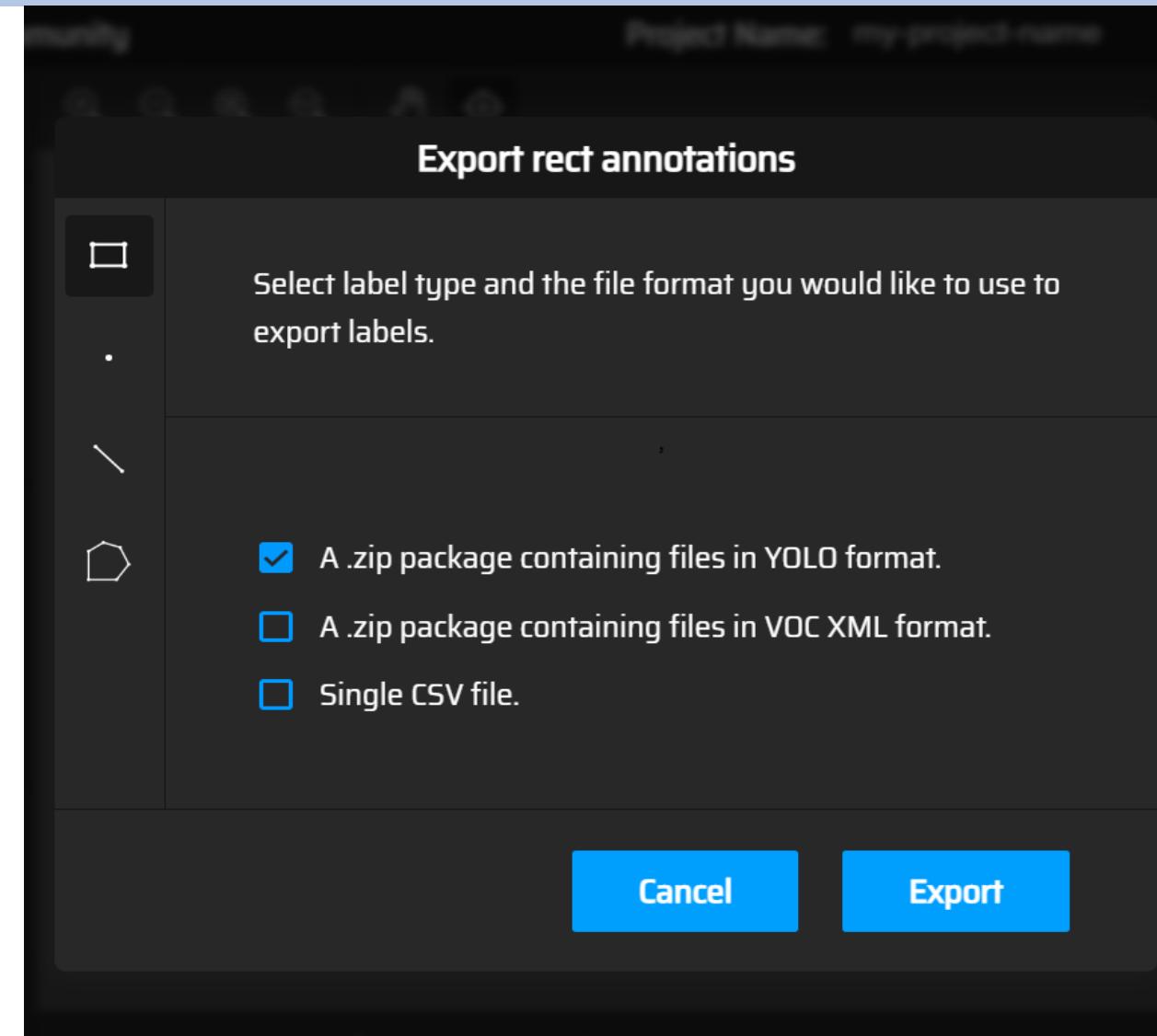
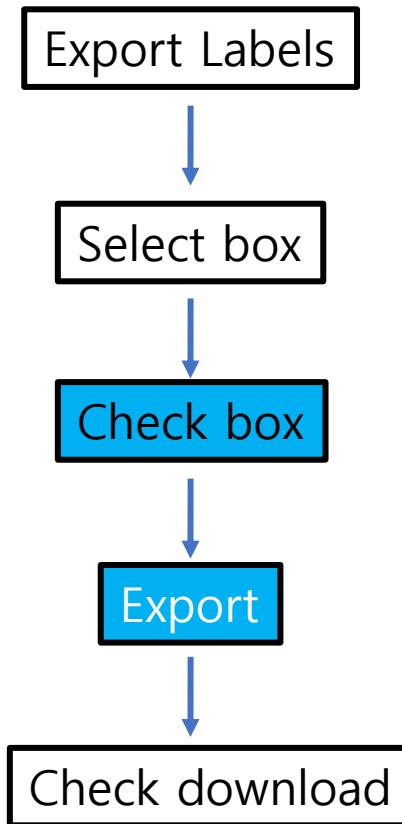
Data preparation : Make Your Own Dataset



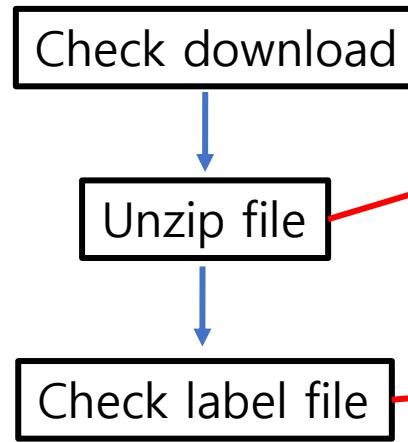
Data preparation : Make Your Own Dataset



Data preparation : Make Your Own Dataset



Data preparation : Make Your Own Dataset



이름	수정한 날짜	유형	크기
1.txt	2023-04-28 ...	텍스트 문서	1KB
2.txt	2023-04-28 ...	텍스트 문서	1KB
3.txt	2023-04-28 ...	텍스트 문서	1KB
4.txt	2023-04-28 ...	텍스트 문서	1KB
5.txt	2023-04-28 ...	텍스트 문서	1KB
6.txt	2023-04-28 ...	텍스트 문서	1KB
7.txt	2023-04-28 ...	텍스트 문서	1KB
8.txt	2023-04-28 ...	텍스트 문서	1KB
9.txt	2023-04-28 ...	텍스트 문서	1KB
10.txt	2023-04-28 ...	텍스트 문서	1KB
11.txt	2023-04-28 ...	텍스트 문서	1KB
12.txt	2023-04-28 ...	텍스트 문서	1KB
13.txt	2023-04-28 ...	텍스트 문서	1KB
14.txt	2023-04-28 ...	텍스트 문서	1KB
15.txt	2023-04-28 ...	텍스트 문서	1KB

Data preparation : Make Your Own Dataset

Check download

Unzip file

Check label file



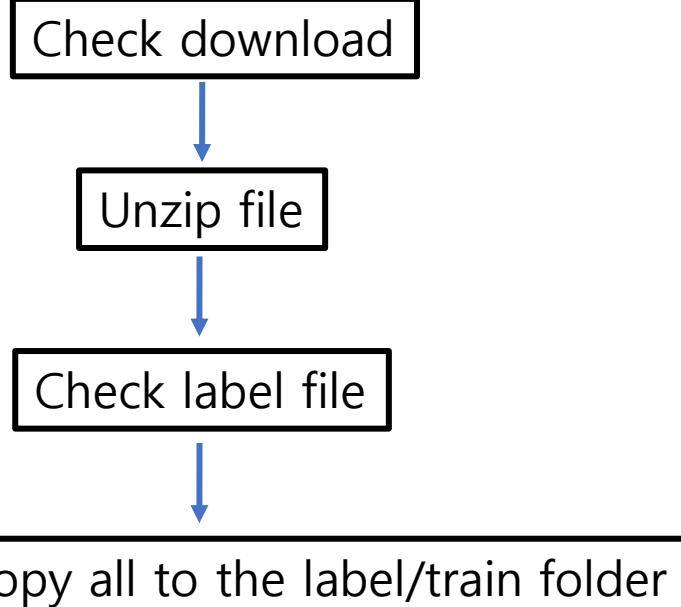
5.txt

편집 보기

```
1 0.331030 0.500044 0.429769 0.999912  
0 0.648641 0.726765 0.264151 0.535989
```

이름	수정한 날짜	유형	크기
1.txt	2023-04-28 ...	텍스트 문서	1KB
2.txt	2023-04-28 ...	텍스트 문서	1KB
3.txt	2023-04-28 ...	텍스트 문서	1KB
4.txt	2023-04-28 ...	텍스트 문서	1KB
5.txt	2023-04-28 ...	텍스트 문서	1KB
6.txt	2023-04-28 ...	텍스트 문서	1KB
7.txt	2023-04-28 ...	텍스트 문서	1KB
8.txt	2023-04-28 ...	텍스트 문서	1KB
9.txt	2023-04-28 ...	텍스트 문서	1KB
10.txt	2023-04-28 ...	텍스트 문서	1KB
11.txt	2023-04-28 ...	텍스트 문서	1KB
12.txt	2023-04-28 ...	텍스트 문서	1KB
13.txt	2023-04-28 ...	텍스트 문서	1KB
14.txt	2023-04-28 ...	텍스트 문서	1KB
15.txt	2023-04-28 ...	텍스트 문서	1KB

Data preparation : Make Your Own Dataset



내 PC > 다운로드 > labels_my-project-name_2023-04-28-12-27-52			
이름	수정한 날짜	유형	크기
1.txt	2023-04-28 ...	텍스트 문서	1KB
2.txt	2023-04-28 ...	텍스트 문서	1KB
3.txt	2023-04-28 ...	텍스트 문서	1KB
4.txt	2023-04-28 ...	텍스트 문서	1KB
5.txt	2023-04-28 ...	텍스트 문서	1KB
6.txt	2023-04-28 ...	텍스트 문서	1KB
7.txt	2023-04-28 ...	텍스트 문서	1KB
8.txt	2023-04-28 ...	텍스트 문서	1KB
9.txt	2023-04-28 ...	텍스트 문서	1KB
10.txt	2023-04-28 ...	텍스트 문서	1KB
11.txt	2023-04-28 ...	텍스트 문서	1KB
12.txt	2023-04-28 ...	텍스트 문서	1KB
13.txt	2023-04-28 ...	텍스트 문서	1KB
14.txt	2023-04-28 ...	텍스트 문서	1KB
15.txt	2023-04-28 ...	텍스트 문서	1KB

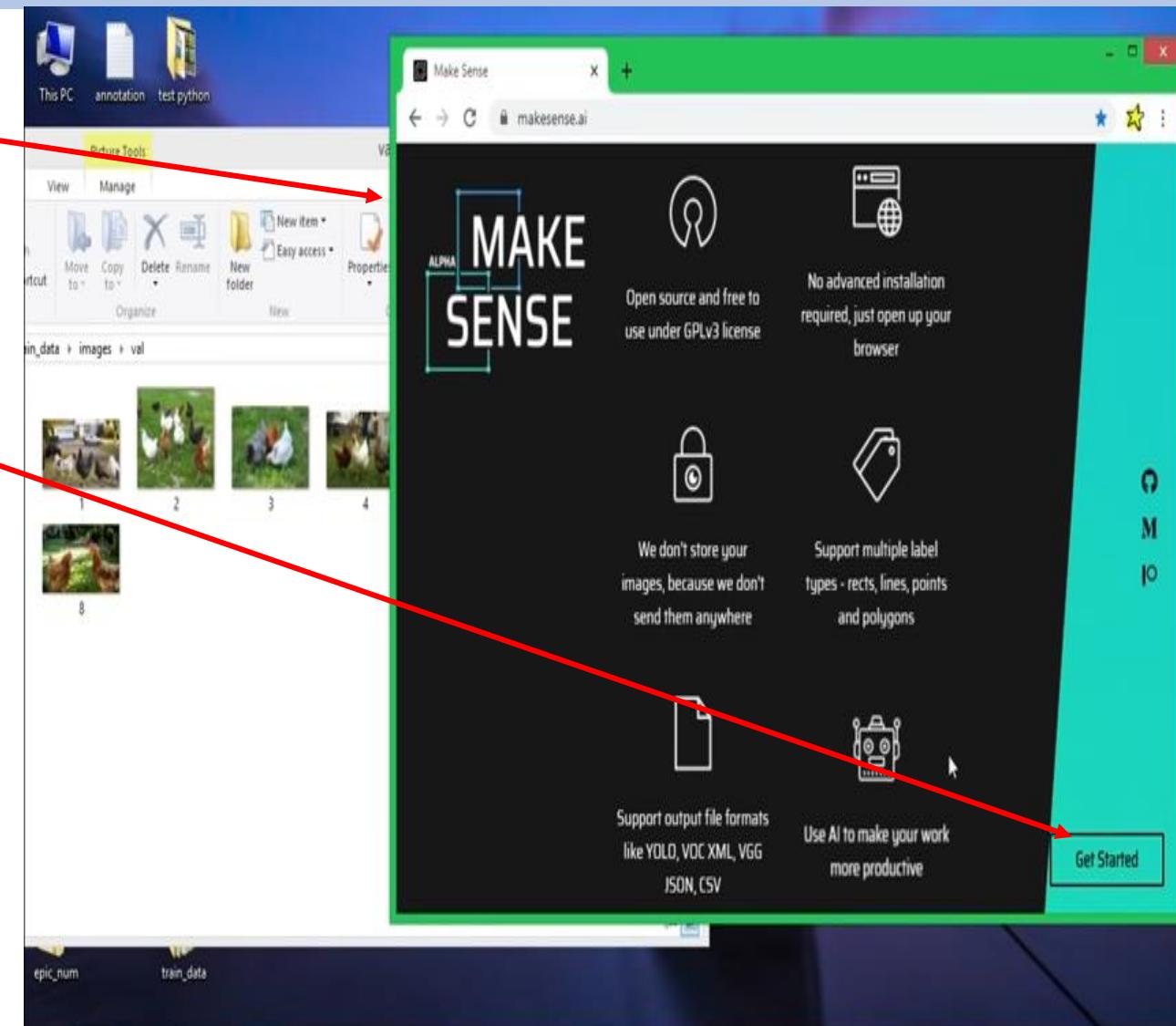
Data preparation : Make Your Own Dataset

Do same proc for val

Go to makesense.ai for data labeling

Get Started

<https://github.com/SkalskiP/make-sense>



Data preparation : Make Your Own Dataset

Go to makesense.ai for data labeling



Get Started



Drag Drop Images

Drag val folder in images/train

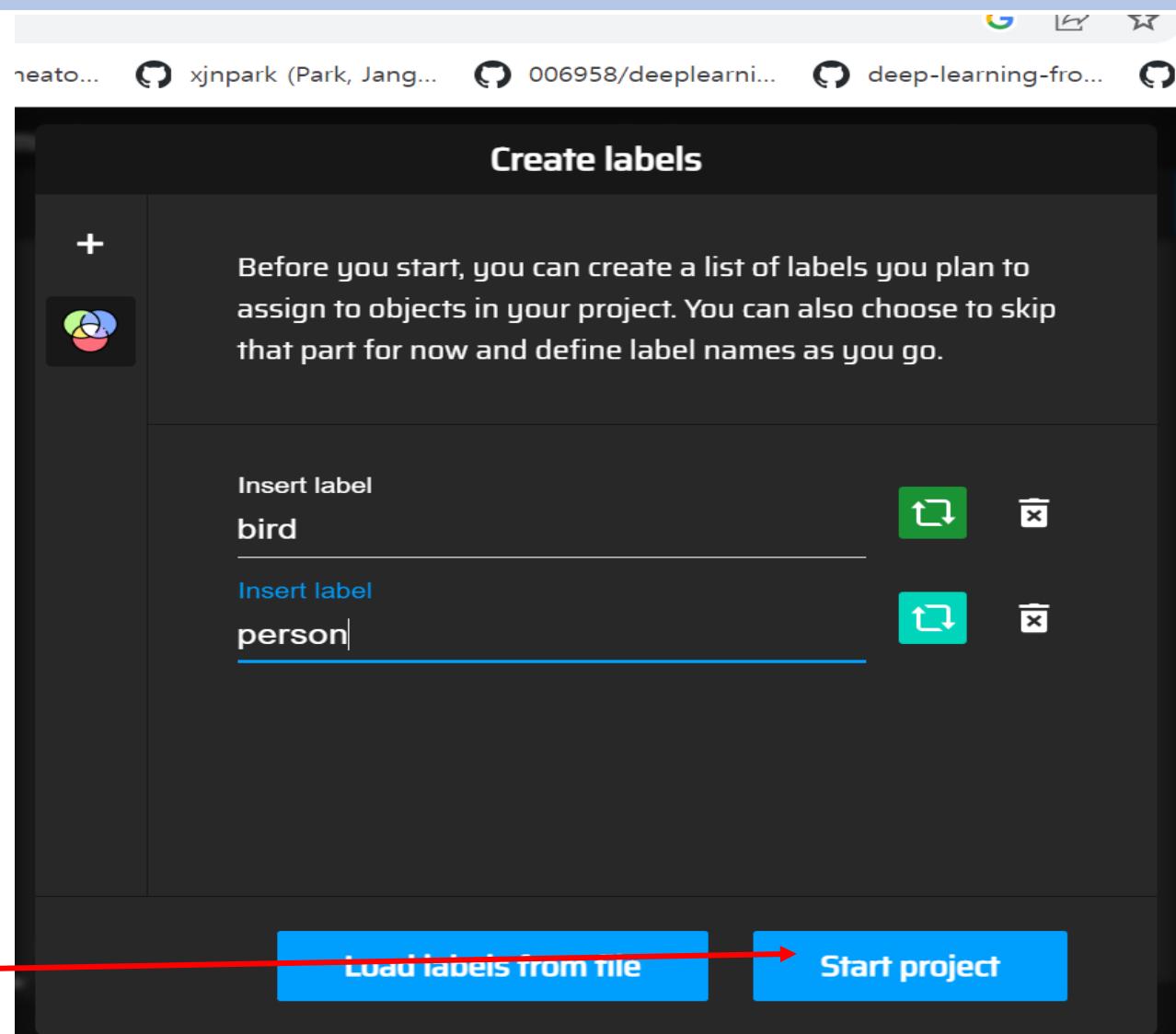


Create Labels

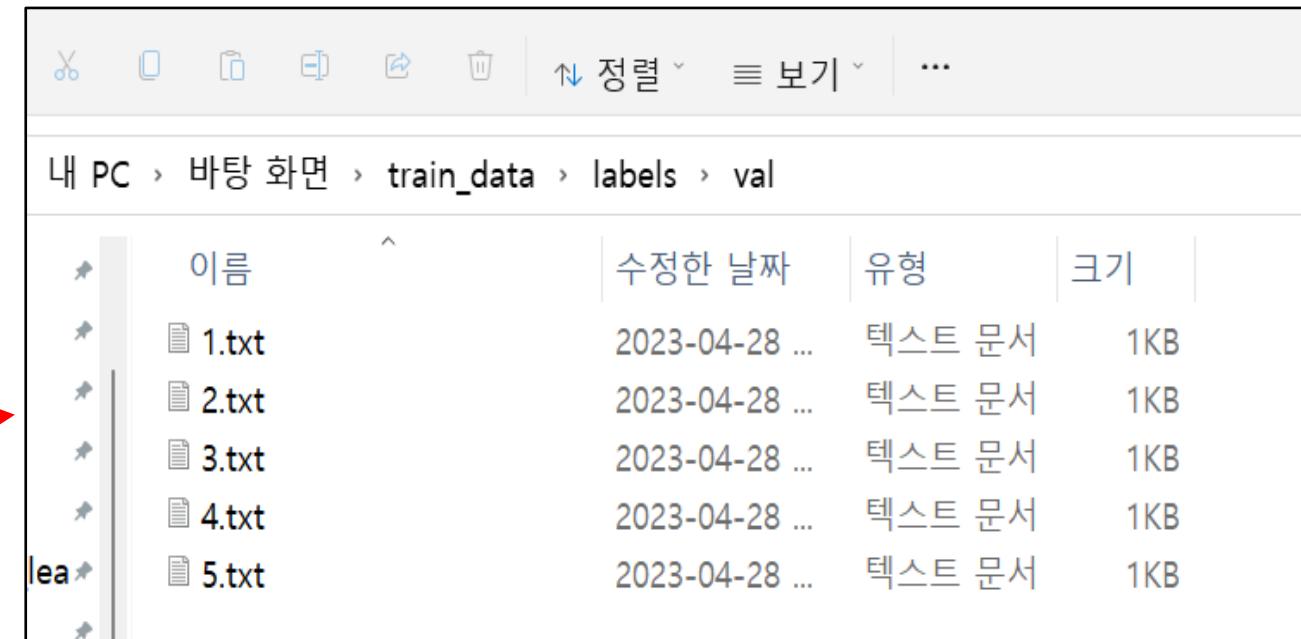
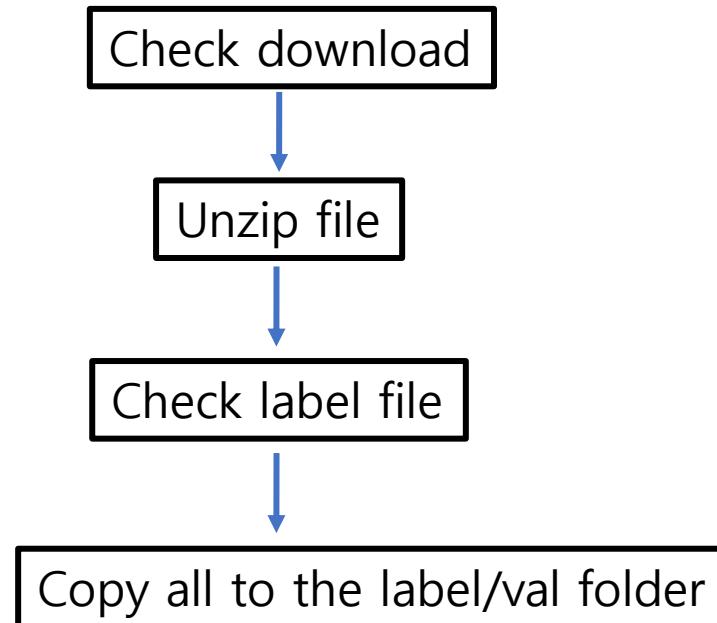
Click + or enter, class1 name : bird

Click + or enter, class2 name : person

Start project



Data preparation : Make Your Own Dataset



Now, we ready for training

Run YOLOv5 in Colab

Training

Go to : <https://github.com/ultralytics/yolov5>

Open in Colab

Run 1st cell

Check folders

ultralytics YOLOv5 v7.0

English | 简体中文

YOLOv5 CI failing DOI 10.5281/zenodo.7347926 docker pulls 279k

Run on Gradient Open in Colab Open in Kaggle

YOLOv5 Tutorial

File Edit View Insert Runtime Tools Help Cannot save changes

Files

yolov5

classify

data

models

segment

utils

CITATION.cff

CONTRIBUTING.md

LICENSE

README

README.zh-CN.md

benchmarks.py

detect.py

export.py

hubconf.py

requirements.txt

1 !git clone https://github.com/ultralytics/yolov5 # clone
2 %cd yolov5
3 %pip install -qr requirements.txt # install
4
5 import torch
6 import utils
7 display = utils.notebook_init() # checks

YOLOv5 v7.0-155-g8ecc727 Python-3.10.11 torch-2.0.0+cu118 CUDA-11.7 Setup complete (2 CPUs, 12.7 GB RAM, 23.3/78.2 GB disk)

Run YOLOv5 in Colab

Training

Go to : <https://github.com/ultralytics/yolov5>



Open in Colab



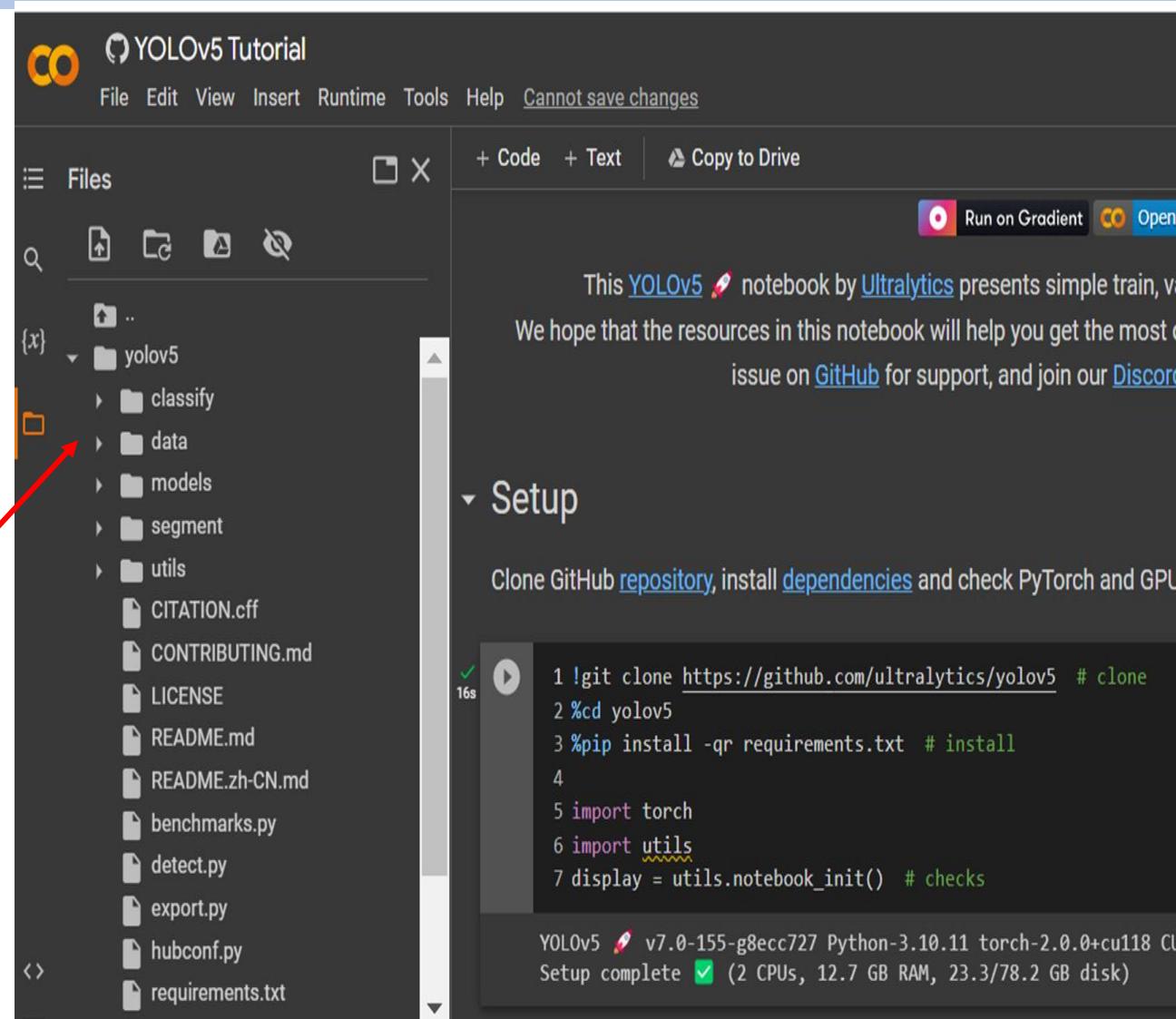
Run 1st cell



Check folders



Data/coco128.yaml



CO YOLOv5 Tutorial

File Edit View Insert Runtime Tools Help Cannot save changes

Files

+ Code + Text Copy to Drive

Run on Gradient Open

This YOLOv5 🚀 notebook by Ultralytics presents simple train, val, test and deploy steps.

We hope that the resources in this notebook will help you get the most out of YOLOv5.

issue on GitHub for support, and join our Discord.

Setup

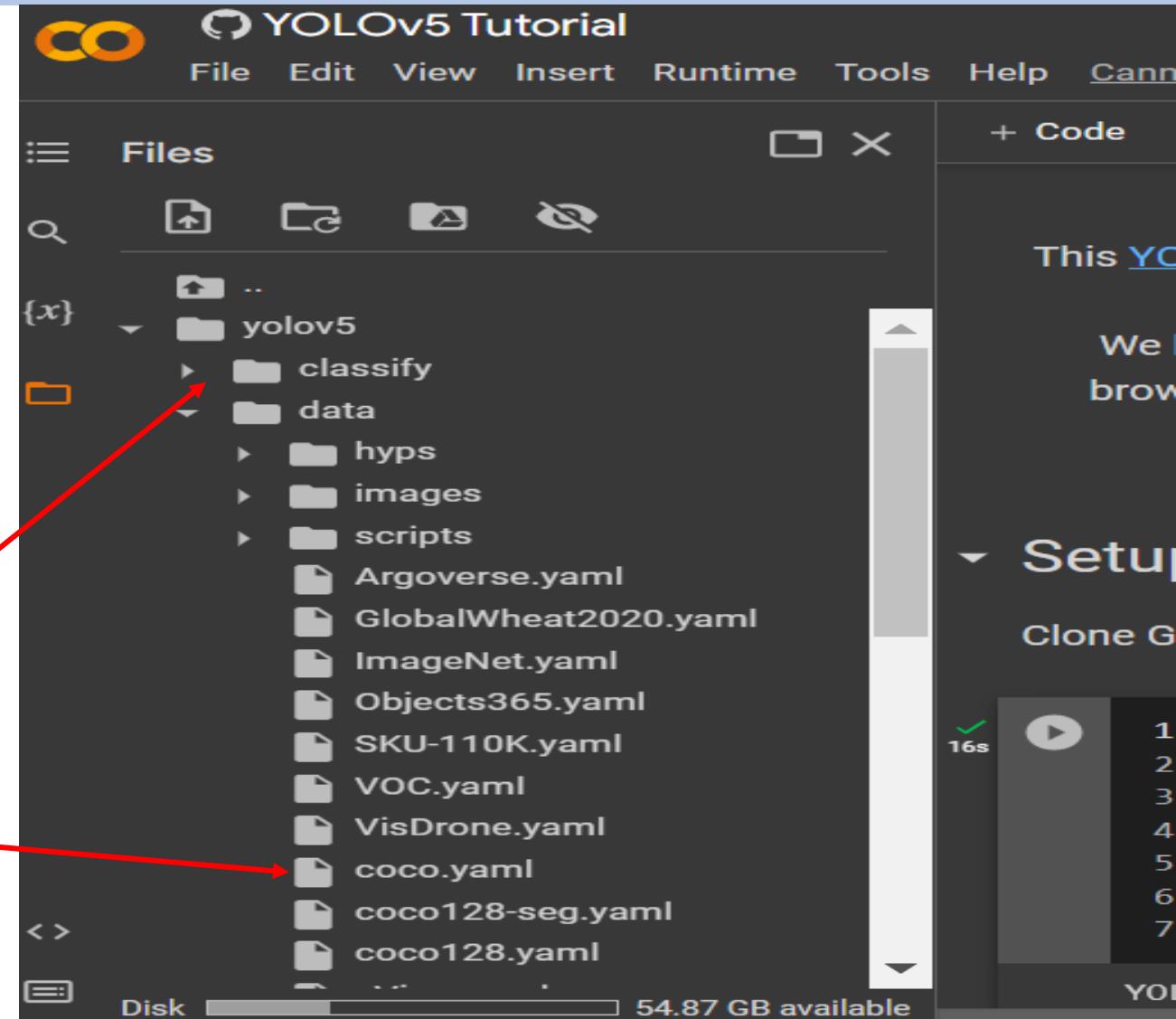
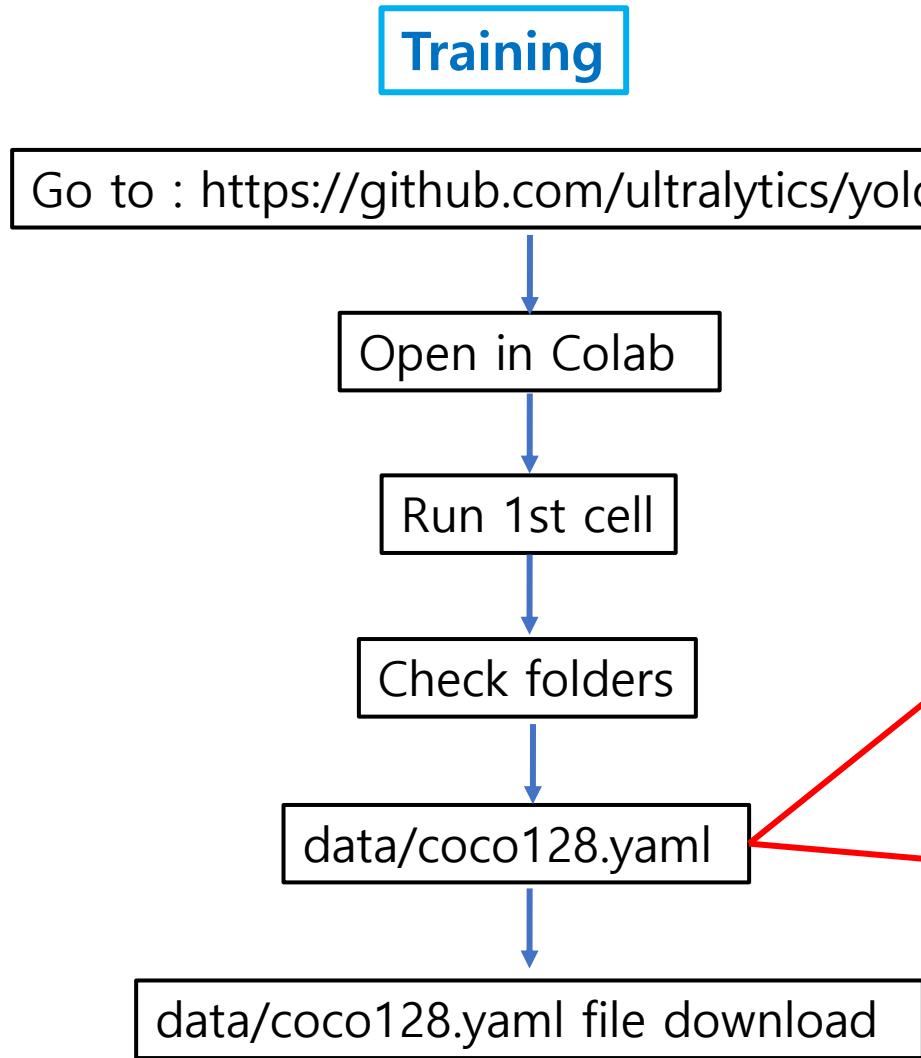
Clone GitHub repository, install dependencies and check PyTorch and GPU compatibility.

```
1 !git clone https://github.com/ultralytics/yolov5 # clone
2 %cd yolov5
3 %pip install -qr requirements.txt # install
4
5 import torch
6 import utils
7 display = utils.notebook_init() # checks
```

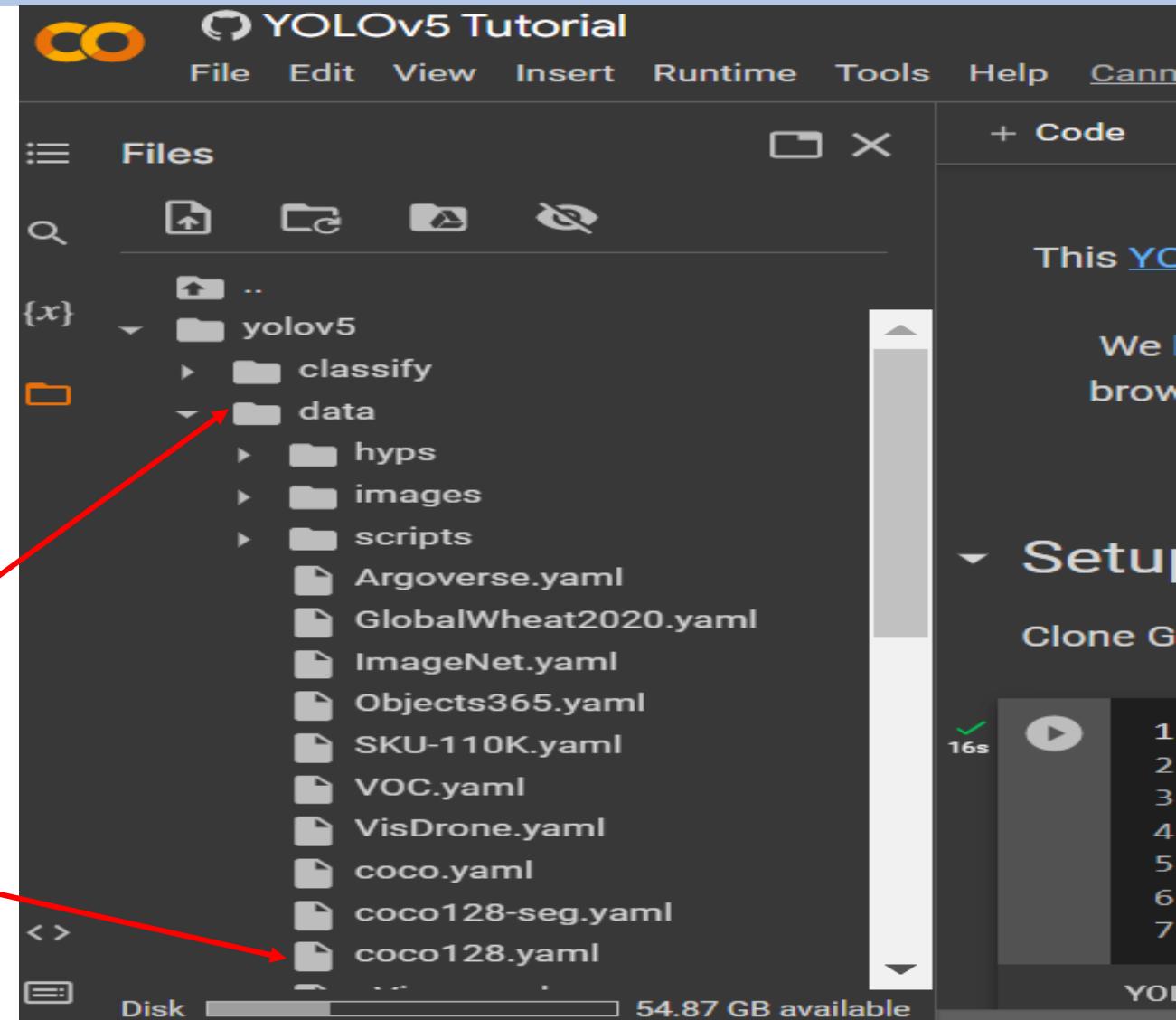
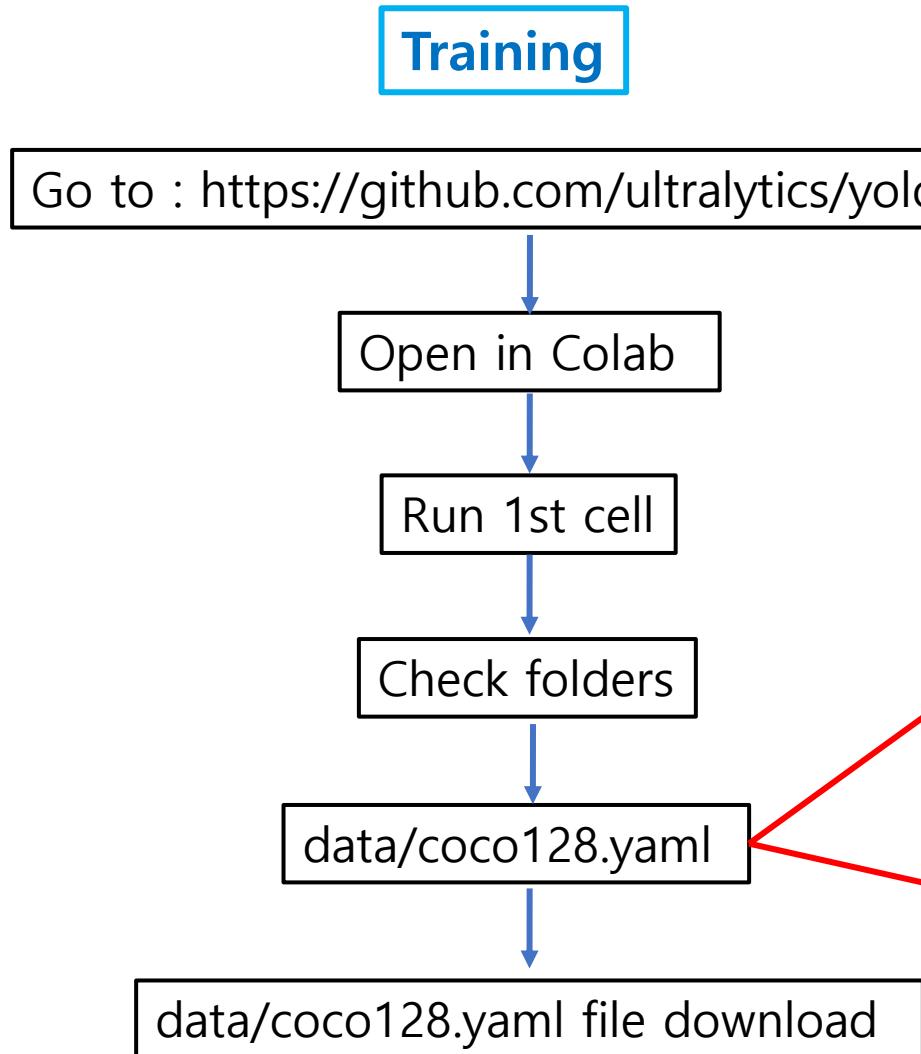
YOLOv5 🚀 v7.0-155-g8ecc727 Python-3.10.11 torch-2.0.0+cu118 CUDA-11.8

Setup complete ✓ (2 CPUs, 12.7 GB RAM, 23.3/78.2 GB disk)

Run YOLOv5 in Colab



Run YOLOv5 in Colab



Run YOLOv5 in Colab

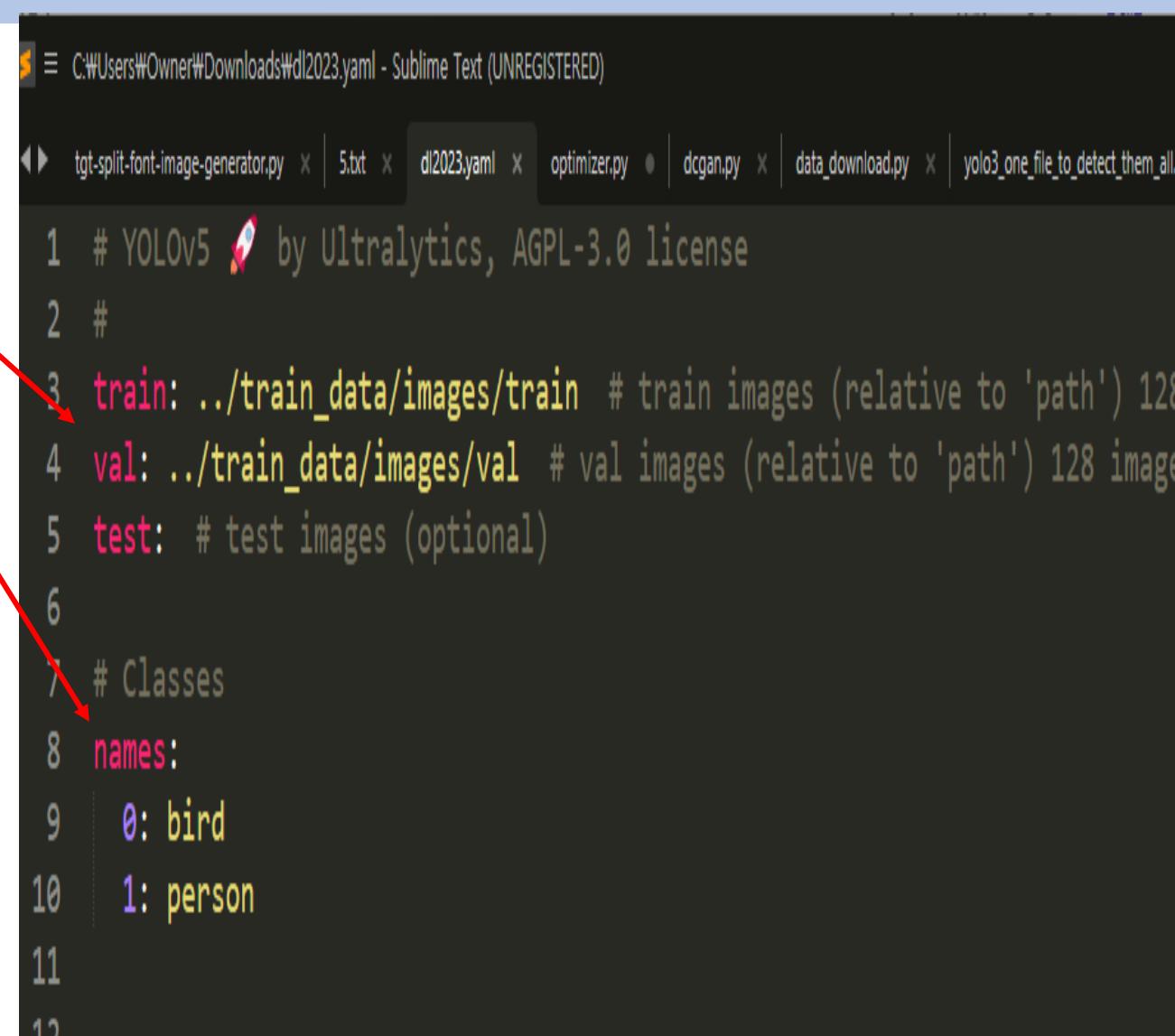
Edit coco128.yaml file and save as
your own name

```
C:\Users\Owner\Downloads\coco128.yaml - Sublime Text (UNREGISTERED)
tgt-split-font-image-generator.py | 5.txt | coco128.yaml | optimizer.py | dcgan.py | data_download.py | yolo

1 # YOLOv5 🚀 by Ultralytics, AGPL-3.0 license
2 # COCO128 dataset https://www.kaggle.com/ultralytics/coco128
3 # Example usage: python train.py --data coco128.yaml
4 # parent
5 #   └── yolov5
6 #     └── datasets
7 #       └── coco128 ← downloads here (7 MB)
8
9
10 # Train/val/test sets as 1) dir: path/to/imgs, 2) file: path/
11 path: ../datasets/coco128 # dataset root dir
12 train: images/train2017 # train images (relative to 'path')
13 val: images/val2017 # val images (relative to 'path') 128
14 test: # test images (optional)
15
16 # Classes
17 names:
18   0: person
19   1: bicycle
20   2: car
21   3: motorcycle
22   4: airplane
23   5: bus
24   6: train
25   7: truck
26   8: boat
27   9: traffic light
28  10: fire hydrant
29  11: stop sign
30  12: parking meter
31  13: bench
32  14: hand
```

Run YOLOv5 in Colab

Edit coco128.yaml file and save it as your own name



```
C:\Users\Owner\Downloads\dl2023.yaml - Sublime Text (UNREGISTERED)
tgt-split-font-image-generator.py x 5.txt x dl2023.yaml optimizer.py ● dcgan.py x data_download.py x yolo3_one_file_to_detect_them_all.py x

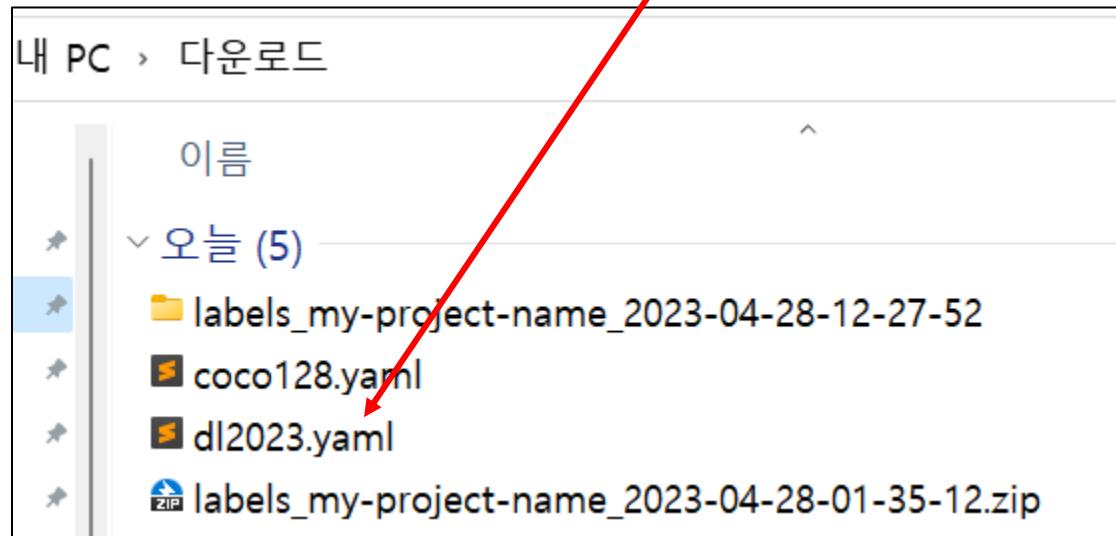
1 # YOLOv5 🚀 by Ultralytics, AGPL-3.0 license
2 #
3 train: ../train_data/images/train # train images (relative to 'path') 128 images
4 val: ../train_data/images/val # val images (relative to 'path') 128 images
5 test: # test images (optional)
6
7 # Classes
8 names:
9 | 0: bird
10 | 1: person
11
12
```

Run YOLOv5 in Colab

Edit coco128.yaml file and save it as your own name



save it as 'dl2023'



```
C:\Users\Owner\Downloads\dl2023.yaml - Sublime Text (UNREGISTERED)

tgt-split-font-image-generator.py x 5.txt x dl2023.yaml optimizer.py ● dgan.py x data_download.py x yolo3_one_file_to_detect_them_all.py

1 # YOLOv5 🚀 by Ultralytics, AGPL-3.0 license
2 #
3 train: ../train_data/images/train # train images (relative to 'path') 128 images
4 val: ../train_data/images/val # val images (relative to 'path') 128 images
5 test: # test images (optional)
6
7 # Classes
8 names:
9 | 0: bird
10 | 1: person
11
12
```

Run YOLOv5 in Colab

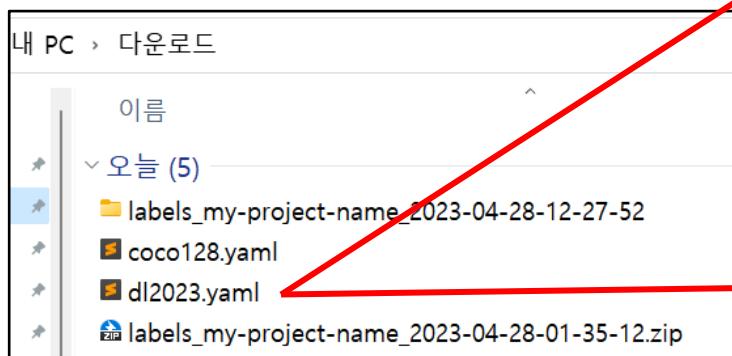
Edit coco128.yaml file and save it as your own name



save it as 'dl2023'



Drag and Upload 'dl2023.yaml' At data folder

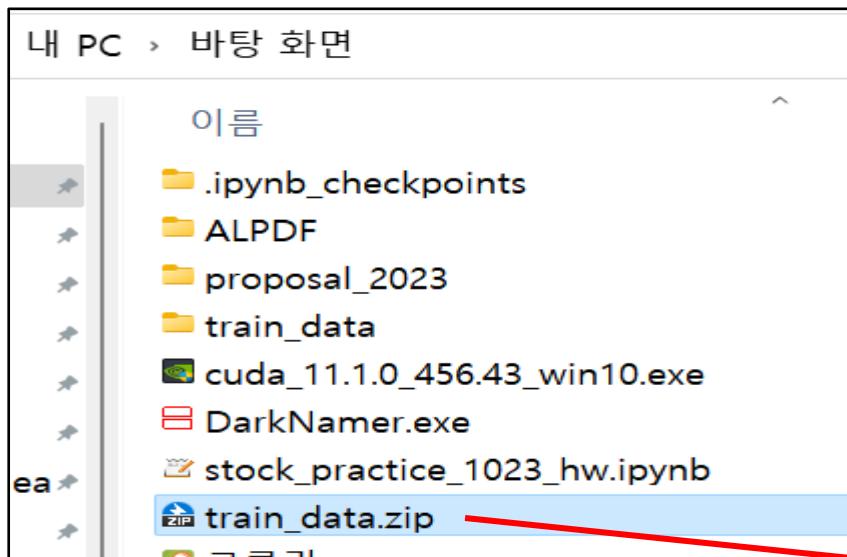


Almost done !!!

Run YOLOv5 in Colab

Make train_data fold to zip file

Drag and in the
Colab folder



YOLOv5 Tutorial

File Edit View Insert Runtime Tools Help Cannot save changes

+ Code + Text Copy to Drive

15s

```
4
5 import torch
6 import utils
7 display = utils.notebook_init()
```

YOLOv5 v7.0-155-g8ecc
Setup complete (2 CPU)

1. Detect

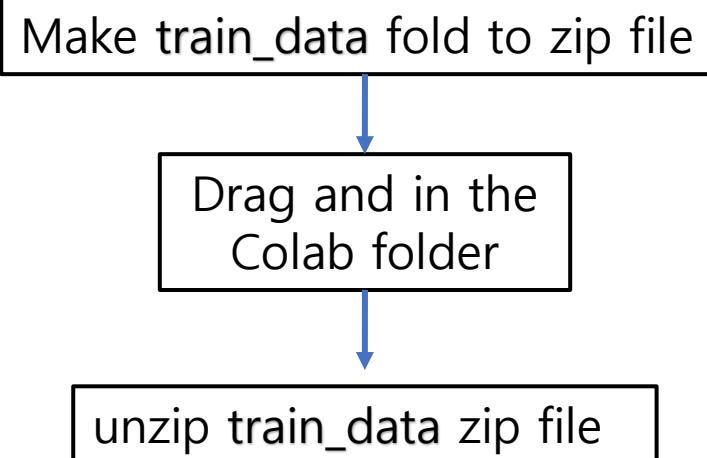
detect.py runs YOLOv5 inference
results to runs/detect. Example

```
python detect.py --source 0 # 웹캠  
img.jpg  
vid.mp4  
screen  
path/  
'path/*'  
'https://  
'rtsp://
```

Disk 34.87 GB available

```
[ ] 1 !python detect.py --weights yolov5s.pt --source 0  
2 # display.Image(filename)
```

Run YOLOv5 in Colab



```
✓ 0s [ ] 1 !unzip -q ../train_data.zip -d ../
```

The screenshot shows the Google Colab interface. On the left, the 'Files' sidebar displays a directory structure: .., train_data, yolov5, and train_data.zip. A red arrow points from the 'train_data.zip' file in the sidebar to the play button in the code cell below. Another red arrow points from the play button in the code cell to the play button in the main toolbar. The code cell contains the command `!unzip -q ../train_data.zip -d ../`. To the right of the code cell is a section titled '1. Detect' which includes a description of the `detect.py` script and its usage examples.

YOLoV5 Tutorial

File Edit View Insert Runtime Tools Help Cannot save changes

Files

.., train_data, yolov5, train_data.zip

+ Code + Text Copy to Drive

[1] 4
5 import torch
6 import utils
7 display = utils.notebook_init() # checks

YOLOv5 v7.0-155-g8ecc727 Python-3.10.11 tor...
Setup complete (2 CPUs, 12.7 GB RAM, 23.3/7

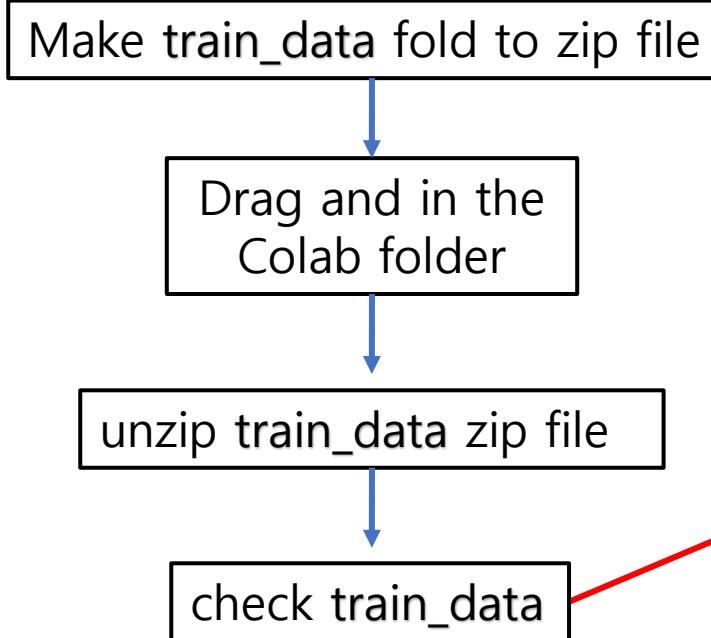
1 !unzip -q ../train_data.zip -d ../

1. Detect

`detect.py` runs YOLOv5 inference on a variety of sources. Results are saved to `runs/detect`. Example inference sources are:

```
python detect.py --source 0 # webcam
                  img.jpg # image
                  vid.mp4 # video
                  screen # screenshot
                  path/ # directory
                  'path/*.jpg' # glob
                  'https://youtu.be/Zgi9g1ksQHc'
                  'rtsp://example.com/media.mp4'
```

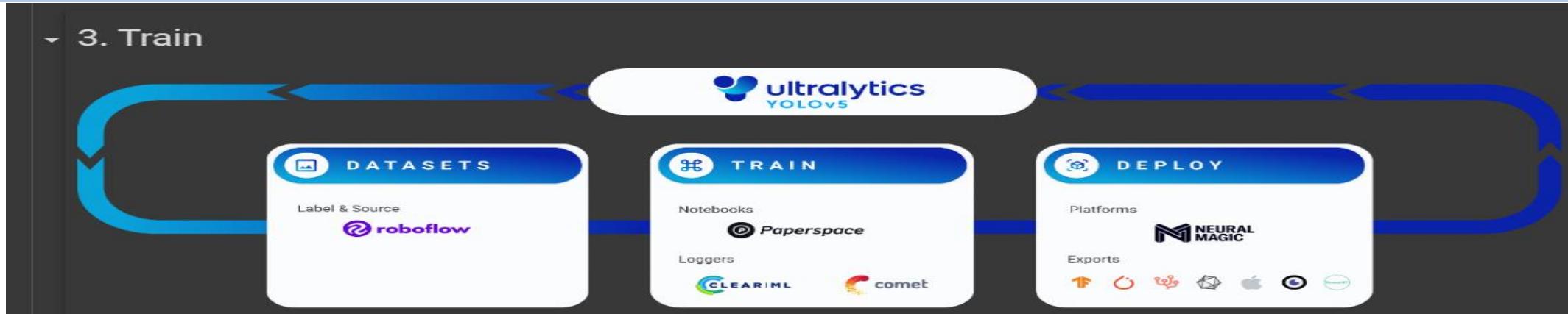
Run YOLOv5 in Colab



The screenshot shows the Google Colab interface. On the left, the "Files" sidebar displays a directory structure: .., train_data (containing images and labels folders, each with train and val subfolders), yolov5, and train_data.zip. A red arrow points from the "check train_data" step in the flowchart to the train_data folder in the sidebar. On the right, a code cell titled "YOLOv5 Tutorial" contains Python code for setting up YOLOv5. The code includes importing torch and utils, displaying notebook initialization, and unzipping the train_data.zip file. Below the code cell, a section titled "1. Detect" provides information about running inference with detect.py. At the bottom, there is a disk usage bar indicating 54.87 GB available.

```
File Edit View Insert Runtime Tools Help Cannot save changes + Code + Text Copy to Drive [1] 15s 4 import torch 5 import utils 6 display = utils.notebook_init() # che 1 !unzip -q ../train_data.zip -d ../ 0s 1. Detect detect.py runs YOLOv5 inference on a variety of s results to runs/detect. Example inference source python detect.py --source 0 # webcam img.jpg # image vid.mp4 # video screen # screenshot path/ # directory 'path/*.jpg' # glob 'https://youtu.be/Zgi9g1k... 'rtsp://example.com/media
```

Run YOLOv5 in Colab



```
[ ] 1 # Train YOLOv5s on COCO128 for 3 epochs
2 !python train.py --img 640 --batch 16 --epochs 3 --data coco128.yaml --weights yolov5s.pt --cache
```

Go to Train part → Change command

```
1 # Train YOLOv5s on COCO128 for 3 epochs
2 !python train.py --img 640 --batch 1 --epochs 200 --data dl2023.yaml --weights yolov5s.pt --cache
```

Run YOLOv5 in Colab

CO YOLOv5 Tutorial

File Edit View Insert Runtime Tools Help Cannot save changes

Files

- ..
- train_data
- yolov5
 - classify
 - data
 - hyp
 - images
 - scripts
 - Argoverse.yaml
 - GlobalWheat2020.yaml
 - ImageNet.yaml
 - Objects365.yaml
 - SKU-110K.yaml
 - VOC.yaml
 - VisDrone.yaml
 - coco.yaml
 - coco128-seg.yaml
 - coco128.yaml
 - dl2023.yaml
 - xView.yaml
 - models
 - runs
 - segment

+ Code + Text Copy to Drive

Epoch	GPU_mem	box_loss	obj_loss	cls_loss	Instances	P	R	mAP50	mAP50-95
198/199	0.713G	0.03073	0.02828	0.007342	5	0.933	0.536	0.656	0.389
	Class	Images	Instances						
	all	5	9	0.933	0.536	0.656	0.389		
199/199	0.713G	0.03971	0.03275	0.01019	3	0.933	0.536	0.656	0.389
	Class	Images	Instances						
	all	5	9	0.933	0.536	0.656	0.389		

200 epochs completed in 0.061 hours.
Optimizer stripped from runs/train/exp/weights/last.pt, 14.4MB
Optimizer stripped from runs/train/exp/weights/best.pt, 14.4MB

Validating runs/train/exp/weights/best.pt...
Fusing layers...
Model summary: 157 layers, 7015519 parameters, 0 gradients, 15.8 GFLOPs

Class	Images	Instances	P	R	mAP50	mAP50-95
all	5	9	0.728	0.596	0.638	0.418
bird	5	7	0.707	0.692	0.638	0.382
person	5	2	0.75	0.5	0.638	0.454

Results saved to runs/train/exp

4. Visualize

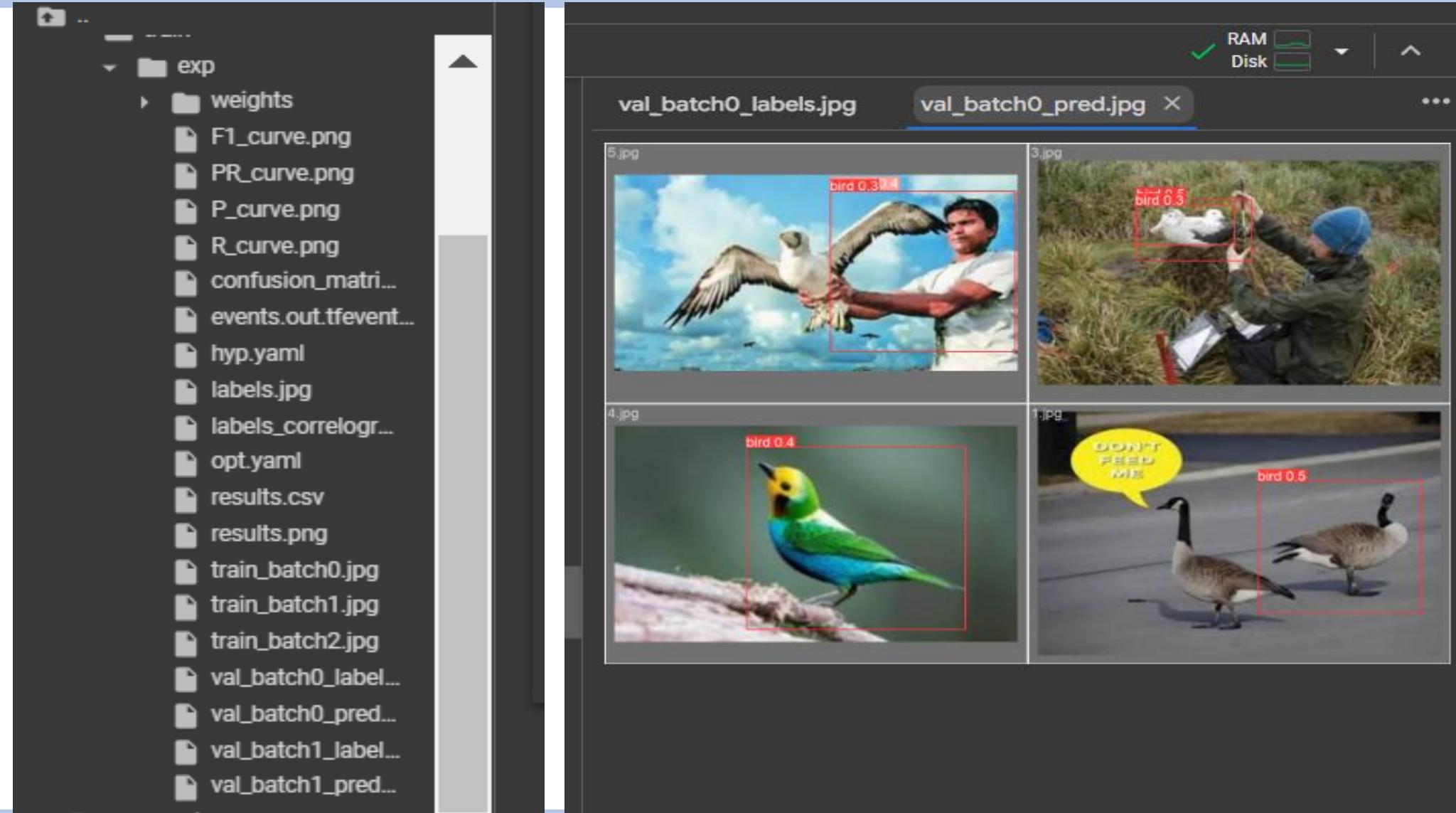
Comet Logging and Visualization NEW

Check result

Run YOLOv5 in Colab

Check result

Download result



Run YOLOv5 in Colab

Check video clip



Check Video Clip

```
[18] 1 !python detect.py --weights yolov5s.pt --img 640 --conf 0.25 --source data/images  
2 # display.Image(filename='runs/detect/exp/zidane.jpg', width=600)
```

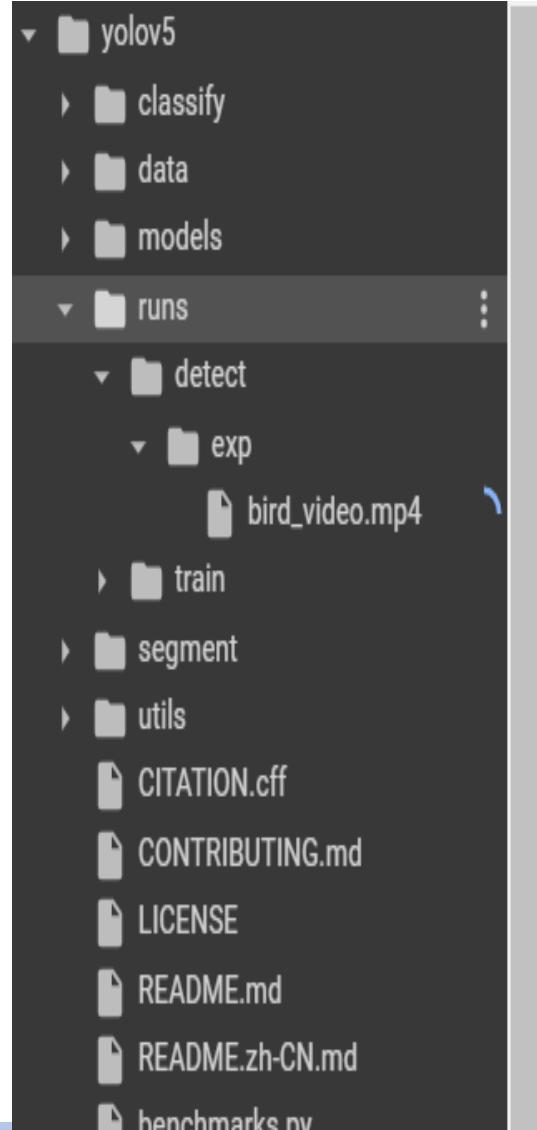
```
1 !python detect.py --weights runs/train/exp/weights/best.pt --img 640 --conf 0.25 --source ..\bird_video.mp4  
2 # display.Image(filename='runs/detect/exp/zidane.jpg', width=600)
```

Validating runs/train/exp/weights/best.pt...
Fusing layers...
Model summary: 157 layers, 7225885 parameters, 0 gradients, 16.4 GFLOPs

Class	Images	Instances	P	R	mAP50	mAP50-95: 100% 4/4 [00:06<00:00, 1.56s/it]
all	128	929	0.759	0.646	0.734	0.49
person	128	254	0.857	0.706	0.805	0.525
bicycle	128	6	0.773	0.577	0.725	0.414
car	128	46	0.664	0.435	0.551	0.24
motorcycle	128	5	0.597	0.3	0.827	0.625



Check Video Clip



Download result

```
video 1/1 (221/240) /content/bird_video.mp4: 640x384 1 bird, 7.6ms
video 1/1 (222/240) /content/bird_video.mp4: 640x384 1 bird, 7.7ms
video 1/1 (223/240) /content/bird_video.mp4: 640x384 1 bird, 7.7ms
video 1/1 (224/240) /content/bird_video.mp4: 640x384 (no detections), 7.6ms
video 1/1 (225/240) /content/bird_video.mp4: 640x384 (no detections), 8.8ms
video 1/1 (226/240) /content/bird_video.mp4: 640x384 1 bird, 7.6ms
video 1/1 (227/240) /content/bird_video.mp4: 640x384 1 bird, 7.7ms
video 1/1 (228/240) /content/bird_video.mp4: 640x384 (no detections), 7.6ms
video 1/1 (229/240) /content/bird_video.mp4: 640x384 2 birds, 7.6ms
video 1/1 (230/240) /content/bird_video.mp4: 640x384 1 bird, 7.7ms
video 1/1 (231/240) /content/bird_video.mp4: 640x384 (no detections), 7.7ms
video 1/1 (232/240) /content/bird_video.mp4: 640x384 1 bird, 7.6ms
video 1/1 (233/240) /content/bird_video.mp4: 640x384 1 bird, 7.6ms
video 1/1 (234/240) /content/bird_video.mp4: 640x384 1 bird, 7.6ms
video 1/1 (235/240) /content/bird_video.mp4: 640x384 1 bird, 7.7ms
video 1/1 (236/240) /content/bird_video.mp4: 640x384 1 bird, 7.7ms
video 1/1 (237/240) /content/bird_video.mp4: 640x384 2 birds, 7.6ms
video 1/1 (238/240) /content/bird_video.mp4: 640x384 2 birds, 7.6ms
video 1/1 (239/240) /content/bird_video.mp4: 640x384 1 bird, 9.2ms
video 1/1 (240/240) /content/bird_video.mp4: 640x384 1 bird, 7.6ms
Speed: 0.5ms pre-process, 8.0ms inference, 1.3ms NMS per image at shape (1, 3, 640, 640)
```

Results saved to **runs/detect/exp**

Check Video Clip

bird 0.59

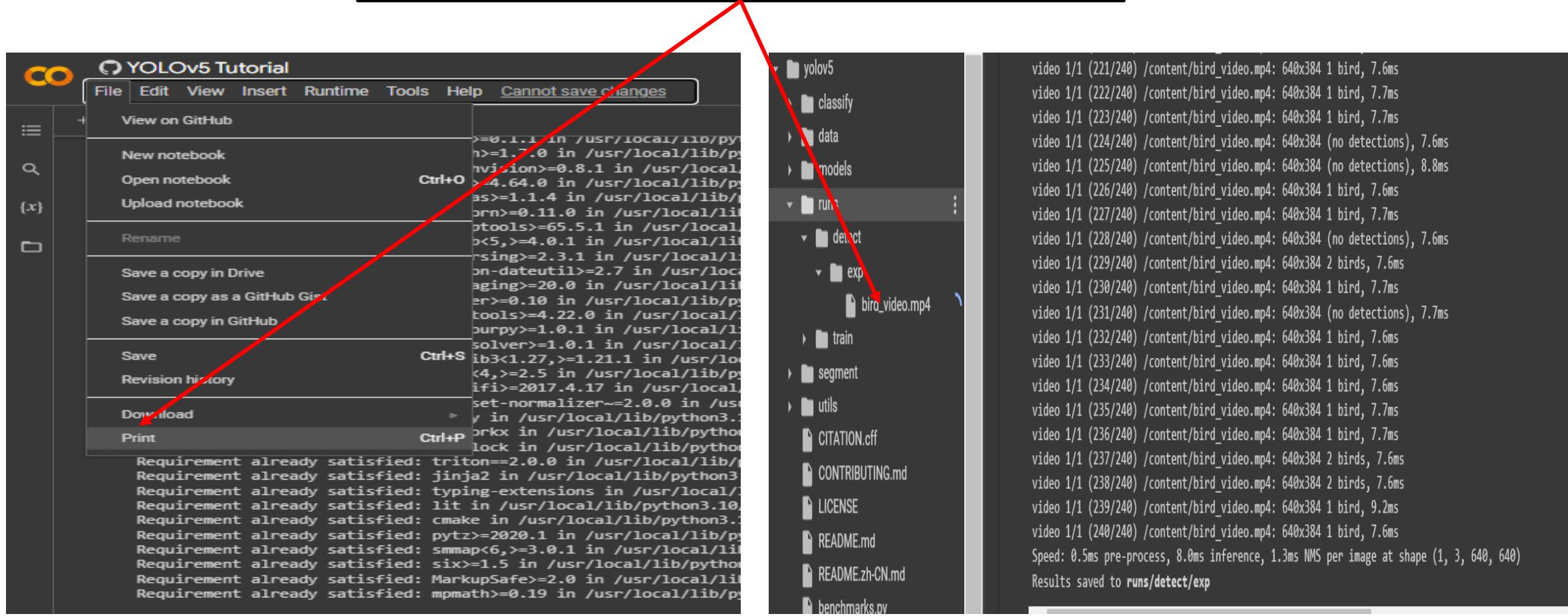


bird 0.74

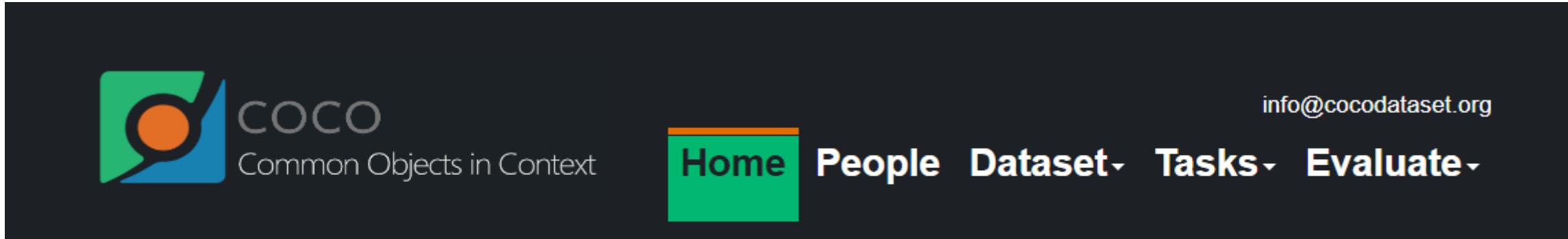


Upload PDF and Video Clip .mp4 file

Print as PDF and upload it
And also upload video clip result you downloaded

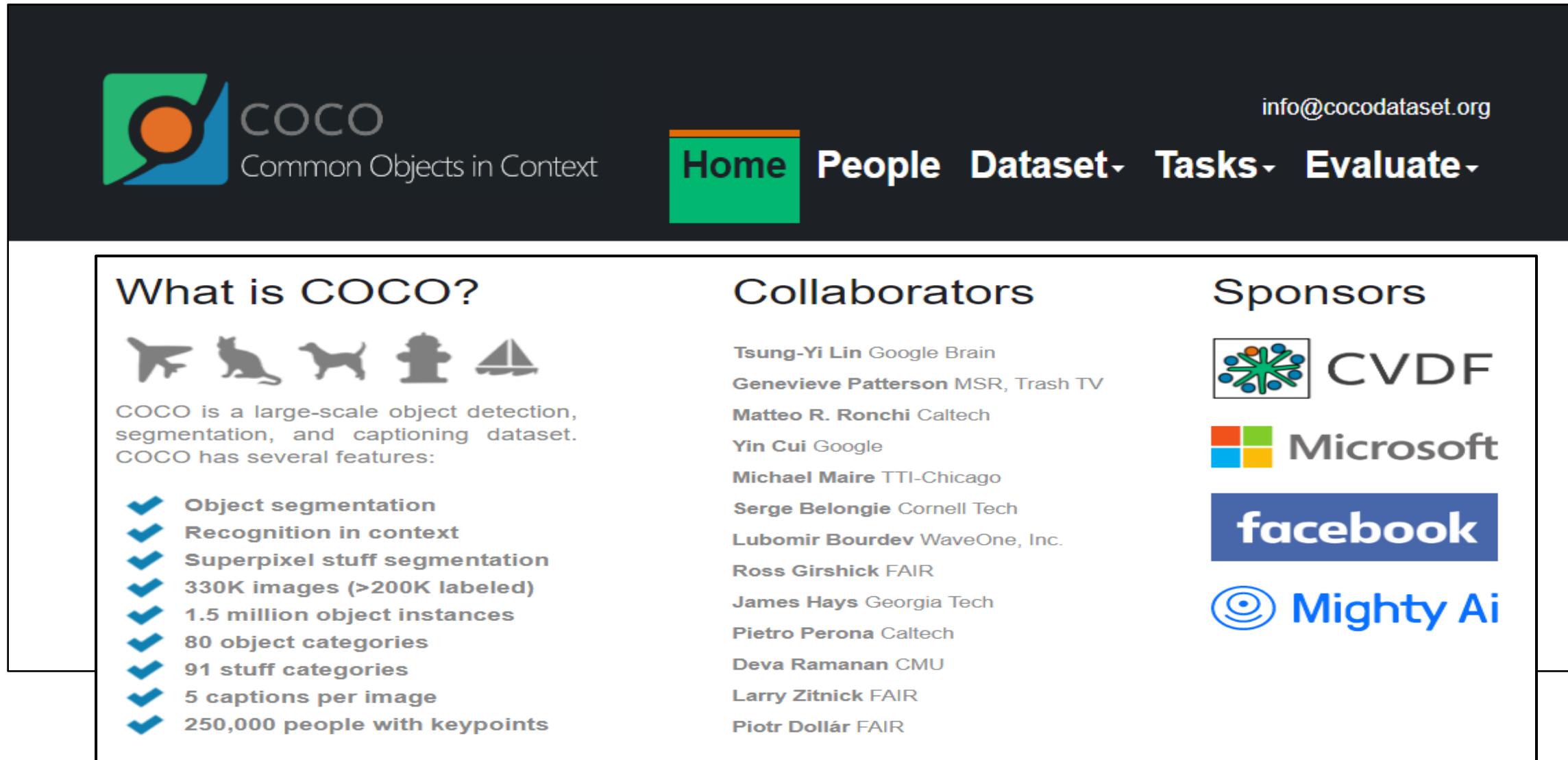


Data Preparation : Using Dataset from sites



News

- We are pleased to announce the [LVIS 2021 Challenge and Workshop](#) to be held at ICCV.
- Please note that there will not be a COCO 2021 Challenge, instead, we encourage people to participate in the LVIS 2021 Challenge.
- We have partnered with the team behind the open-source tool [FiftyOne](#) to make it easier to download, visualize, and evaluate COCO
- [FiftyOne](#) is an open-source tool facilitating visualization and access to COCO data resources and serves as an evaluation tool for model analysis on COCO.



The screenshot shows the COCO dataset homepage. The header features the COCO logo (a stylized 'C' in green, blue, and orange) and the text "COCO Common Objects in Context". On the right, there is an email address "info@cocodataset.org". Below the header, a navigation bar includes "Home" (highlighted in green), "People", "Dataset", "Tasks", and "Evaluate". The main content area has three columns: "What is COCO?", "Collaborators", and "Sponsors".

What is COCO?

COCO is a large-scale object detection, segmentation, and captioning dataset. COCO has several features:

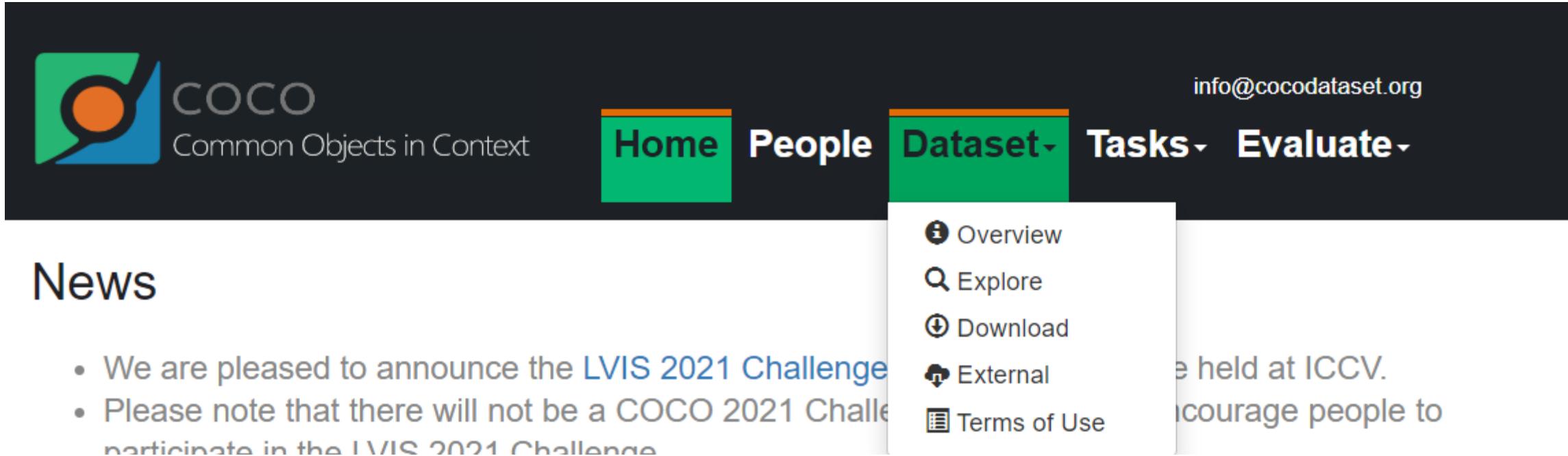
- ✓ Object segmentation
- ✓ Recognition in context
- ✓ Superpixel stuff segmentation
- ✓ 330K images (>200K labeled)
- ✓ 1.5 million object instances
- ✓ 80 object categories
- ✓ 91 stuff categories
- ✓ 5 captions per image
- ✓ 250,000 people with keypoints

Collaborators

Tsung-Yi Lin Google Brain
Genevieve Patterson MSR, Trash TV
Matteo R. Ronchi Caltech
Yin Cui Google
Michael Maire TTI-Chicago
Serge Belongie Cornell Tech
Lubomir Bourdev WaveOne, Inc.
Ross Girshick FAIR
James Hays Georgia Tech
Pietro Perona Caltech
Deva Ramanan CMU
Larry Zitnick FAIR
Piotr Dollár FAIR

Sponsors

 CVDF
 Microsoft
 facebook
 Mighty Ai



The screenshot shows the COCO dataset homepage. The logo 'COCO Common Objects in Context' is on the left. The navigation bar includes 'Home' (highlighted in green), 'People', 'Dataset-' (with a dropdown menu open), 'Tasks-', and 'Evaluate-'. The 'Dataset-' menu contains: Overview (info icon), Explore (magnifying glass icon), Download (down arrow icon), External (cloud icon), and Terms of Use (document icon). On the right, an email address 'info@cocodataset.org' is listed. Below the navigation bar, there's a 'News' section with two bullet points:

- We are pleased to announce the [LVIS 2021 Challenge](#)
- Please note that there will not be a COCO 2021 Challenge. Instead, we encourage people to participate in the LVIS 2021 Challenge.



New Tools

COCO API

- \
- {
- ,

Images

[2014 Train images \[83K/13GB\]](#)
[2014 Val images \[41K/6GB\]](#)
[2014 Test images \[41K/6GB\]](#)
[2015 Test images \[81K/12GB\]](#)
[2017 Train images \[118K/18GB\]](#)
[2017 Val images \[5K/1GB\]](#)
[2017 Test images \[41K/6GB\]](#)
[2017 Unlabeled images \[123K/19GB\]](#)

Annotations

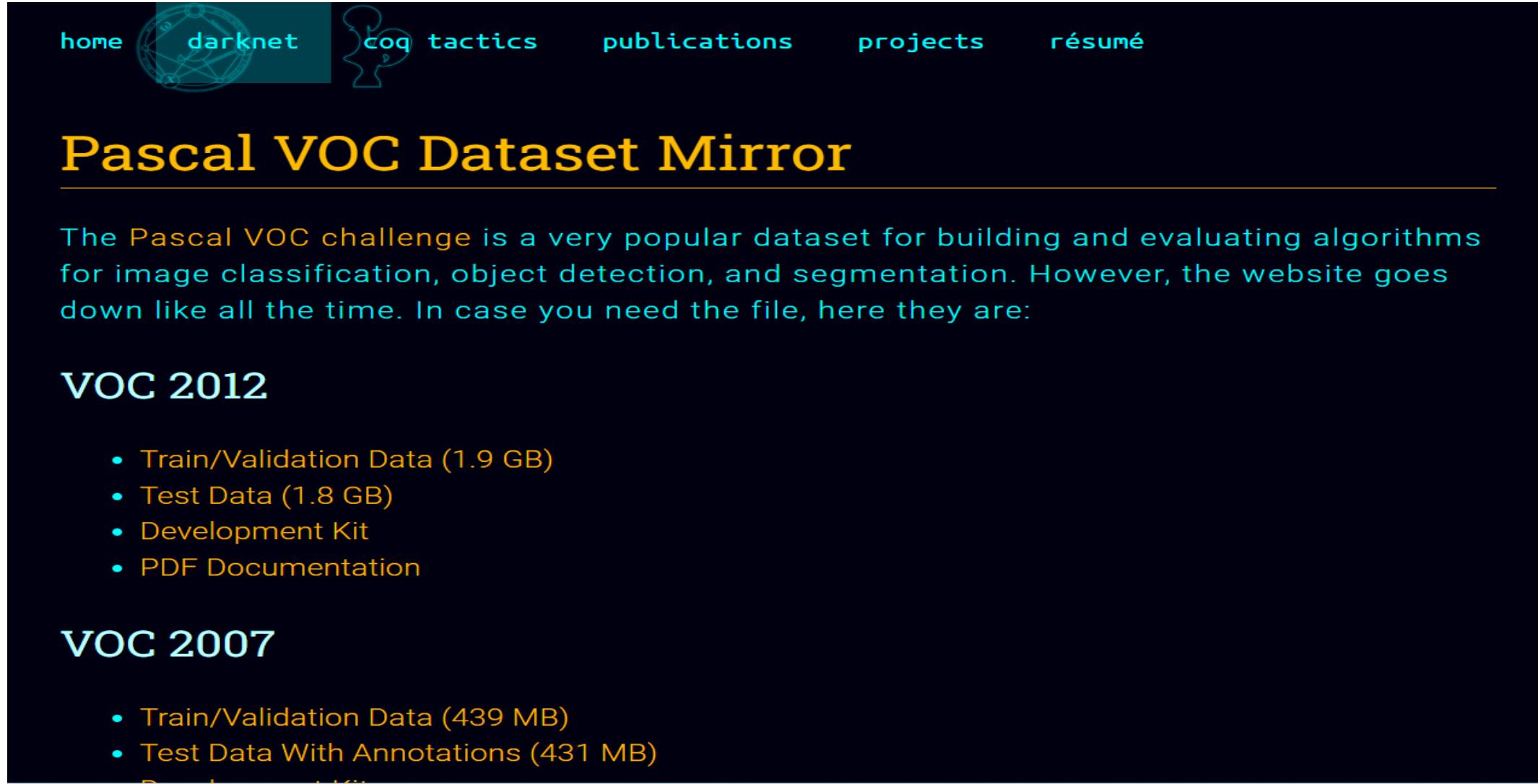
[2014 Train/Val annotations \[241MB\]](#)
[2014 Testing Image info \[1MB\]](#)
[2015 Testing Image info \[2MB\]](#)
[2017 Train/Val annotations \[241MB\]](#)
[2017 Stuff Train/Val annotations \[1.1GB\]](#)
[2017 Panoptic Train/Val annotations \[821MB\]](#)
[2017 Testing Image info \[1MB\]](#)
[2017 Unlabeled Image info \[4MB\]](#)

1. Overview

Which dataset splits should you download? Each year's images are associated with different tasks. Specifically:

2014 Train/Val
2014 Testing
2015 Testing

Detection 2015, Captioning 2015, Detection 2016, Keypoints 2016,
DensePose 2020
Captioning 2015
Detection 2015, Detection 2016, Keypoints 2016
Detection 2017, Keypoints 2017, Stuff 2017,
Detection 2018, Keypoints 2018, Stuff 2018, Panoptic 2018



The screenshot shows a dark-themed website for a dataset mirror. At the top, there's a navigation bar with links: home, darknet, coq tactics, publications, projects, and résumé. Below the navigation is a large yellow header with the text "Pascal VOC Dataset Mirror". Underneath the header, there's a paragraph of text: "The Pascal VOC challenge is a very popular dataset for building and evaluating algorithms for image classification, object detection, and segmentation. However, the website goes down like all the time. In case you need the file, here they are:". Below this text, there are two sections: "VOC 2012" and "VOC 2007", each followed by a bulleted list of files.

VOC 2012

- Train/Validation Data (1.9 GB)
- Test Data (1.8 GB)
- Development Kit
- PDF Documentation

VOC 2007

- Train/Validation Data (439 MB)
- Test Data With Annotations (431 MB)

GENERAL

The 10 Best Public Datasets for Object Detection in 2022

by Clemens Viernickel on September 14th, 2022

[The 10 Best Public Datasets for Object Detection in 2022 | Blog | Scale AI](#)

COMPUTER VISION

50+ Object Detection Datasets from different industry domains

Oct 10, 2020



Last Updated on October 10, 2020 by [Editorial Team](#)

[50+ Object Detection Datasets from different industry domains – Towards AI](#)

Computer Vision Datasets

xView: xView is one of the most massive publicly available datasets of overhead imagery. It contains images from complex scenes around the world, annotated using bounding boxes.

ImageNet: The largest image dataset for computer vision. It provides an accessible image database that is organized hierarchically, according to WordNet.

Kinetics-700: A large-scale dataset of video URLs from Youtube. Including human-centered actions. It contains over 700,000 videos.

Google's Open Images: A vast dataset from Google AI containing over 10 million images.

50+ Object Detection Datasets from different industry domains – Towards AI

Computer Vision Datasets

[xView](#): xView is one of the most massive publicly available datasets of overhead images in the world, announced in 2014. It contains over 5 million labeled images collected from IMDB and Wikipedia.

[ImageNet](#):

accessible via WordNet.

[Color Detection Dataset](#): The dataset contains a CSV file that has 865 color names with their corresponding RGB(red, green, and blue) values of the color. It also has the hexadecimal value of the color.

[Kinetics-70](#):

human-centered

[Google's Open Images](#):

million images

[Stanford Dogs Dataset](#): It contains 20,580 images and 120 different dog breed categories.

Stanford Dogs Dataset

Summary:

- 120 dog breeds
- ~150 images per class
- Total images: 20,580

[Download dataset](#)

[Affenpinscher](#)
(150 images)

ImageNet synset: [n02110627](#)

[Afghan hound](#)
(239 images)

ImageNet synset: [n02088094](#)

[African hunting dog](#)
(169 images)

Stanford Dogs Dataset

[Aditya Khosla](#) [Nityananda Jayadevaprakash](#) [Bangpeng Yao](#) [Li Fei-Fei](#)

Stanford University

The Stanford Dogs dataset contains images of 120 breeds of dogs from around the world. This dataset has been built using images and annotation from ImageNet for the task of fine-grained image categorization. Contents of this dataset:

- **Number of categories:** 120
- **Number of images:** 20,580
- **Annotations:** Class labels, Bounding boxes

Download

You can download the dataset using the links below:

- [Images \(757MB\)](#)
- [Annotations \(21MB\)](#)
- [Lists, with train/test splits \(0.5MB\)](#)
- [Train Features \(1.2GB\), Test Features \(850MB\)](#)
- [README](#)

Data Preparation : Get Dataset from sites

Stanford Dogs Dataset

Summary:

- 120 dog breeds
- ~150 images per class
- Total images: 20,580

[Download dataset](#)

ImageNet synset: [n02113023](#)

Pomeranian
(219 images)

ImageNet synset: [n02112018](#)

Pug
(200 images)

ImageNet synset: [n02110958](#)

Redbone
(148 images)

Pomeranian (219 images)



Data Preparation : Get Dataset from sites

Stanford Dogs Dataset

Summary:

- 120 dog breeds
- ~150 images per class
- Total images: 20,580

[Download dataset](#)

(149 images)

ImageNet synset: [n02086079](#)

[Pembroke](#)

(181 images)

ImageNet synset: [n02113023](#)

[Pomeranian](#)

(219 images)

ImageNet synset: [n02112018](#)

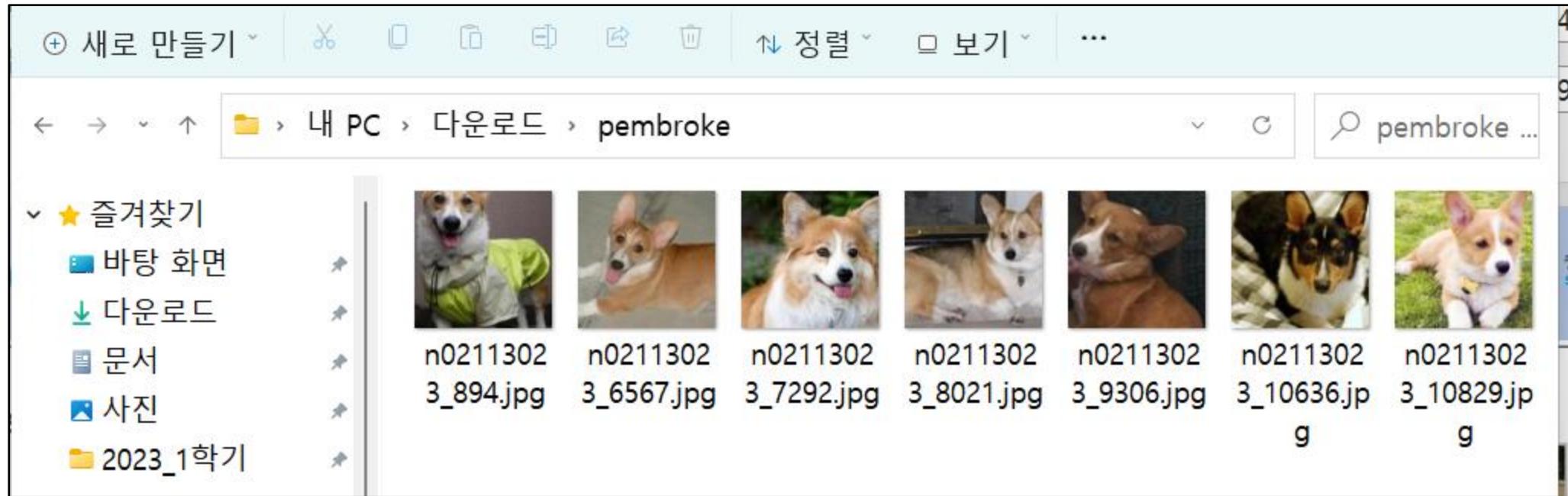
[Pug](#)

(200 images)

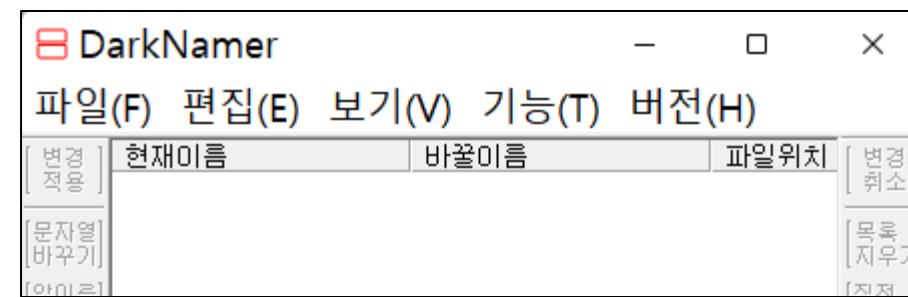
Pembroke (181 images)

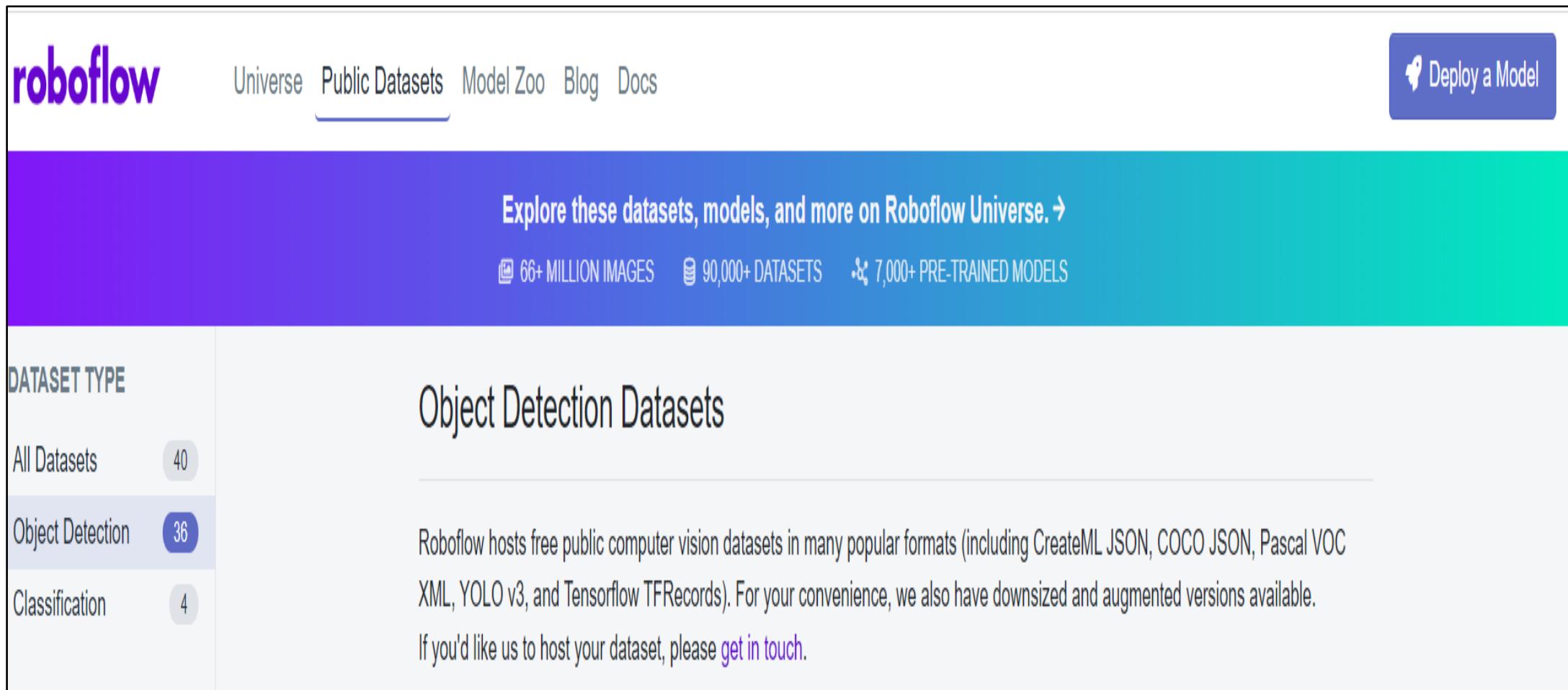


Data Preparation : Get Dataset from sites



✓ Change file name at one time ?





The screenshot shows the Roboflow website's public datasets section. At the top, there is a navigation bar with links: Universe, Public Datasets (which is underlined), Model Zoo, Blog, and Docs. On the right side of the header is a blue button labeled "Deploy a Model". Below the header, a large banner features the text "Explore these datasets, models, and more on Roboflow Universe." with an arrow pointing right, followed by statistics: "66+ MILLION IMAGES", "90,000+ DATASETS", and "7,000+ PRE-TRAINED MODELS". To the left, a sidebar titled "DATASET TYPE" lists categories: All Datasets (40), Object Detection (36, which is highlighted in blue), and Classification (4). The main content area is titled "Object Detection Datasets". It contains a paragraph explaining that Roboflow hosts free public computer vision datasets in various formats like CreateML JSON, COCO JSON, Pascal VOC XML, YOLO v3, and Tensorflow TFRecords, and mentions downsized and augmented versions. It also invites users to contact them for dataset hosting.

Explore these datasets, models, and more on Roboflow Universe. →

66+ MILLION IMAGES 90,000+ DATASETS 7,000+ PRE-TRAINED MODELS

DATASET TYPE

- All Datasets 40
- Object Detection 36
- Classification 4

Object Detection Datasets

Roboflow hosts free public computer vision datasets in many popular formats (including CreateML JSON, COCO JSON, Pascal VOC XML, YOLO v3, and Tensorflow TFRecords). For your convenience, we also have downsized and augmented versions available.

If you'd like us to host your dataset, please [get in touch](#).

<https://public.roboflow.com/object-detection>

Data Preparation : Get Dataset from sites

The screenshot shows the Roboflow Public Datasets page. On the left, there's a sidebar titled "DATASET TYPE" with categories: All Datasets (40), Object Detection (36), and Classification (4). Below the sidebar, the URL <https://pub> is visible. The main content area has a header with the Roboflow logo and navigation links: Universe, Public Datasets (underlined), Model Zoo, Blog, Docs, and a "Deploy a Model" button. A large blue banner at the top right says "Explore these datasets, models, and more on Roboflow Universe. →" followed by statistics: 66+ MILLION IMAGES, 90,000+ DATASETS, and 7,000+ PRE-TRAINED MODELS. Two dataset cards are shown: "Website Screenshots Dataset" (Object Detection (Bounding Box)) with 1206 images, 1 exports, and last updated 8 months ago; and "Pistols Dataset" (Object Detection (Bounding Box)) with 2986 images, 1 exports, and last updated 8 months ago.

Universe Public Datasets Model Zoo Blog Docs Deploy a Model

Explore these datasets, models, and more on Roboflow Universe. →

66+ MILLION IMAGES 90,000+ DATASETS 7,000+ PRE-TRAINED MODELS

WEBSITE SCREENSHOTS DATASET
Object Detection (Bounding Box)
1206 images | 1 exports | Last updated 8 months ago

PISTOLS DATASET
Object Detection (Bounding Box)
2986 images | 1 exports | Last updated 8 months ago

Data Preparation : Get Dataset from sites

roboflow

Universe Public Datasets Model Zoo Blog Docs

Explore these datasets, models, and more

66+ MILLION IMAGES 90,000+ MODELS

Dataset Summary

Dataset Health Check

DOWNLOADS

resize-416x416 2973

Pistols Dataset

Shared By University of Grenada February 2020

License Public Domain More Info

Annotations Guns Object Detection

Downloads

resize-416x416 2973 Images

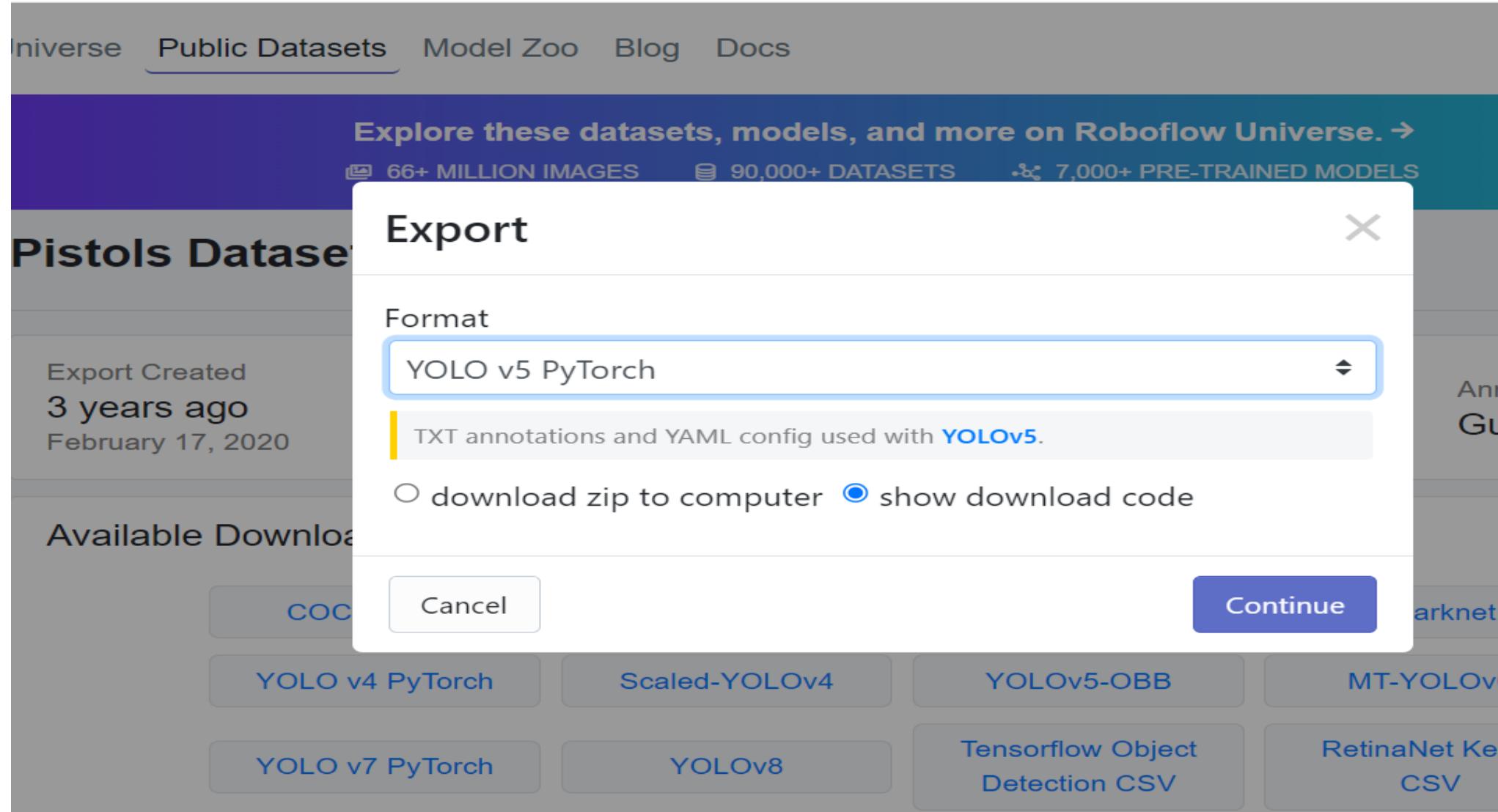
Overview

This dataset contains 2986 images and 3448 labels across a single annotation class: pistols. Images are wide-ranging: pistols in-hand, cartoons, and staged studio quality images of guns.

The dataset was originally released by the [University of Grenada](#), duplicates removed, and rehosted by a Roboflow user.



Data Preparation : Get Dataset from sites



The screenshot shows a modal dialog titled "Export" over a dataset page for "Pistols Dataset". The modal has a dropdown menu set to "YOLO v5 PyTorch". Below it, a note says "TXT annotations and YAML config used with YOLOv5." There are two radio buttons: "download zip to computer" (unchecked) and "show download code" (checked). At the bottom are "Cancel" and "Continue" buttons.

Pistols Dataset

Export Created 3 years ago February 17, 2020

Available Downloads

YOLO v5 PyTorch

YOLO v4 PyTorch Scaled-YOLOv4 YOLOv5-OBB MT-YOLOv6

YOLO v7 PyTorch YOLOv8 Tensorflow Object Detection CSV RetinaNet Keypoint CSV

Explore these datasets, models, and more on Roboflow Universe. →

66+ MILLION IMAGES 90,000+ DATASETS 7,000+ PRE-TRAINED MODELS

Format

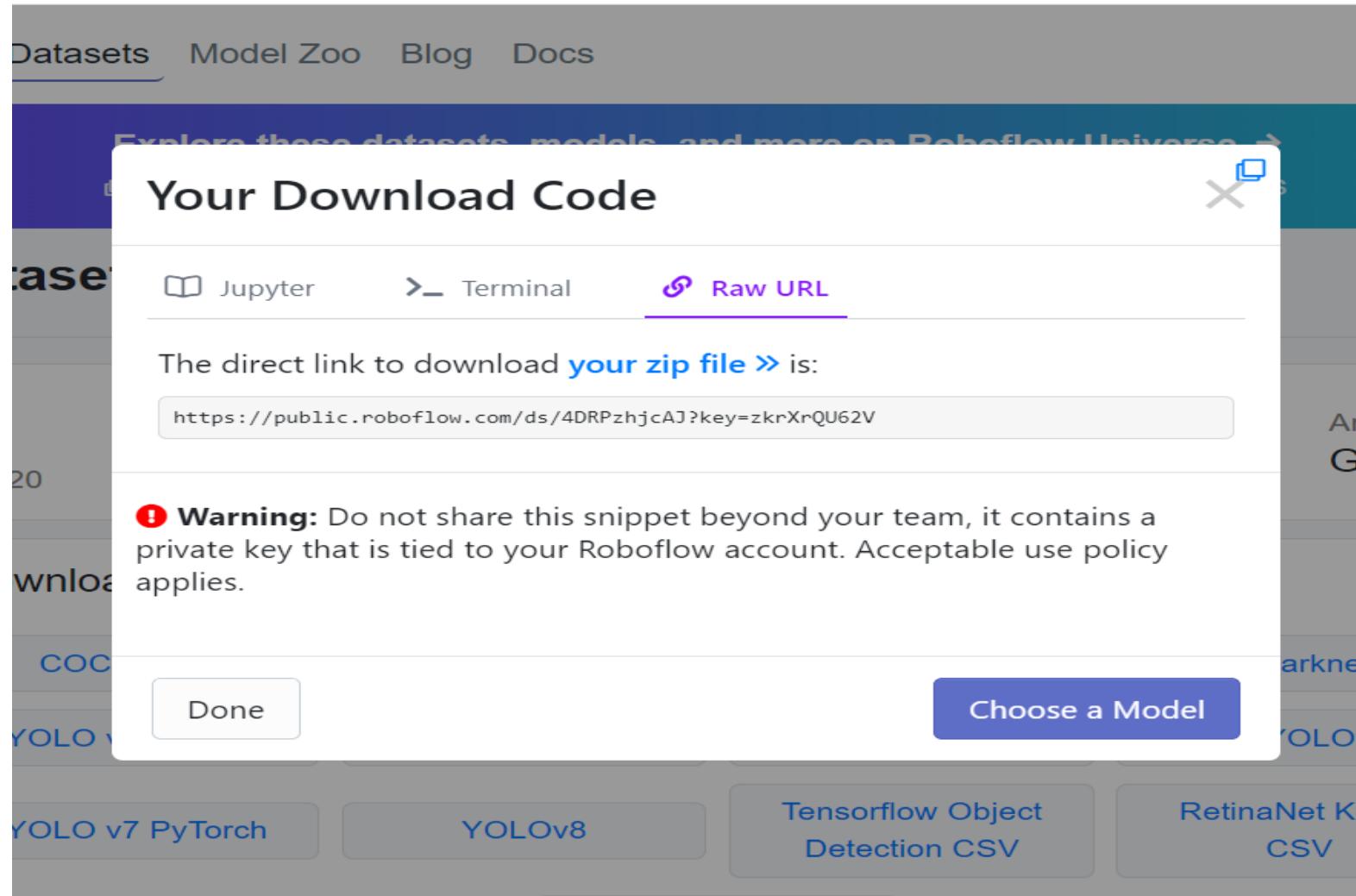
YOLO v5 PyTorch

TXT annotations and YAML config used with YOLOv5.

download zip to computer show download code

Cancel Continue

Data Preparation : Get Dataset from sites



Roboflow Model Library

Next step: Train a model!

The Roboflow Model Library contains pre-configured model architectures for easily training computer vision models. Just add the link to your Roboflow dataset and you're ready to go!

PyTorch Object Detection

YOLOv8

YOLOv8 is a state-of-the-art object detection and image segmentation model created by Ultralytics.

[Tutorial](#) [Video](#) [Repo](#) [Jupyter Notebook](#)

PyTorch Object Detection

YOLOv5

A very fast and easy to use PyTorch model that achieves state of the art (or near state of the art) results.

[Tutorial](#) [Video](#) [Repo](#) [Colab Notebook \(Free GPU\)](#)

[Done](#)

Run YOLOv5 in Colab

The screenshot shows a Google Colab notebook titled "Untitled2.ipynb". The left sidebar displays a file tree with the following structure:

- ..
- export
 - images
 - labels
- sample_data
- README.dataset.txt
- README.roboflow.txt
- data.yaml

The main workspace shows a terminal output window with the following log entries:

```
[1]: extracting: export/labels/armas (874)_jpg.rf.fed7950454
[1]: extracting: export/labels/armas (589)_jpg.rf.fecaabfd1
[1]: extracting: export/labels/armas (295)_jpg.rf.fd3250ald9
[1]: extracting: export/labels/armas (2482)_jpg.rf.fd20edcfa
[1]: extracting: export/labels/armas (22)_jpg.rf.fea65c3366f
[1]: extracting: export/labels/armas (1970)_jpg.rf.fe0b97dd7
[1]: extracting: export/labels/armas (1998)_jpg.rf.fec4e2d07
[1]: extracting: export/labels/armas (2118)_jpg.rf.feaebc3b
[1]: extracting: export/labels/armas (230)_jpg.rf.ff090ac29e
[1]: extracting: export/labels/armas (2490)_jpg.rf.ff618bcee
[1]: extracting: export/labels/armas (2974)_jpg.rf.ff4e6c805
[1]: extracting: export/labels/armas (38)_jpg.rf.ff874c9502a
[1]: extracting: export/labels/armas (948)_jpg.rf.ff128062ce
[1]: extracting: export/labels/armas (339)_jpg.rf.ff862b5e10
[1]: extracting: export/labels/armas (1858)_jpg.rf.ff0769f74
[1]: extracting: export/labels/armas (389)_jpg.rf.ff70678138
[1]: extracting: export/labels/armas (2735)_jpg.rf.ff5f9700c
[1]: extracting: export/labels/armas (1614)_jpg.rf.ffcca44ff
[1]: extracting: data.yaml
[1]: extracting: README.roboflow.txt
[1]: extracting: README.dataset.txt
```

Run YOLOv5 in Colab

Untitled2.ipynb

파일 수정 보기 삽입 렌타임 도구 도움말 모든 변경사항이 저장됨

+ 코드 + 테스트

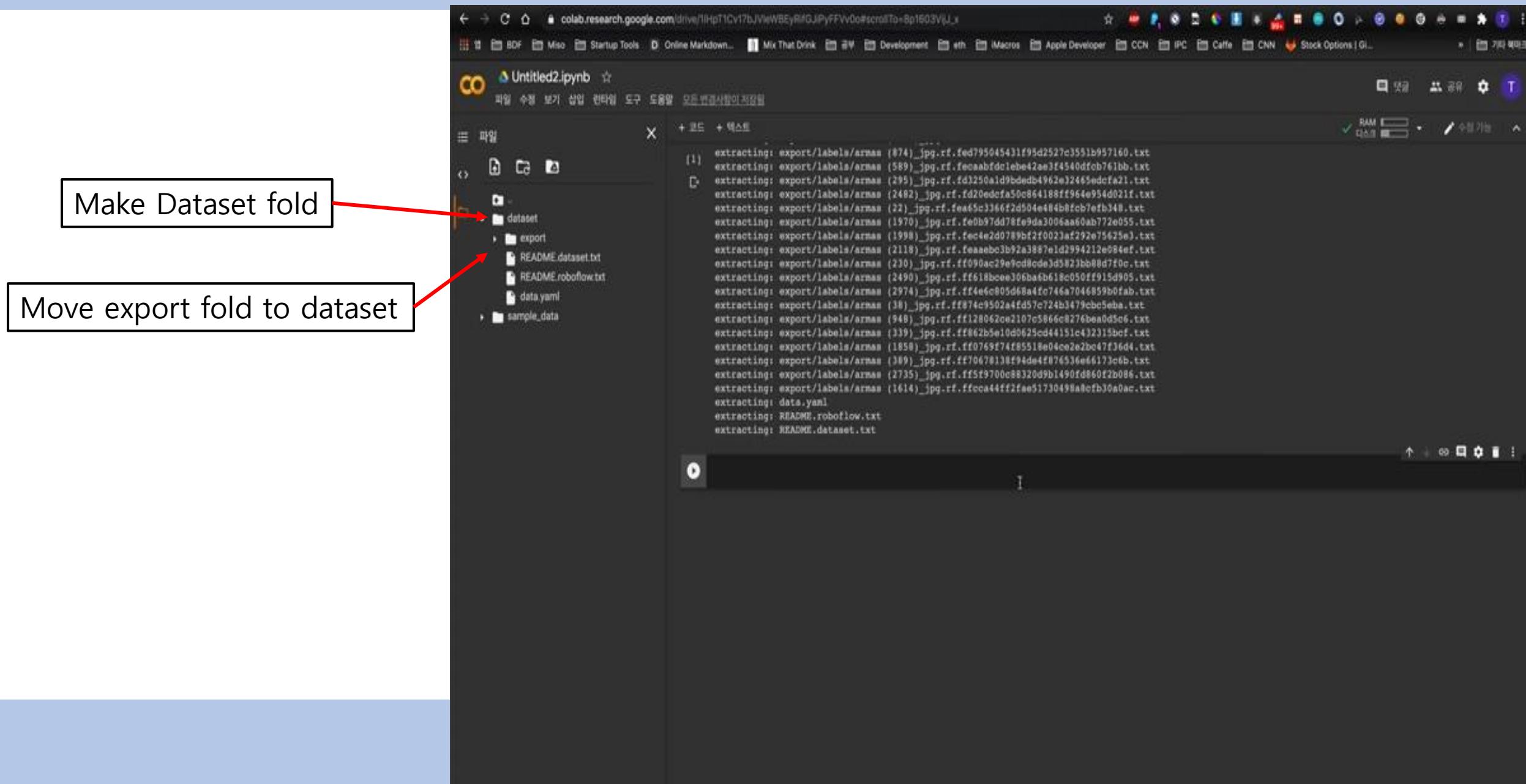
```
(1) extracting: export/labels/armas (874)_jpg_rf.fed795045431f95d2527c3551b957160.txt
extracting: export/labels/armas (589)_jpg_rf.fecaaabfd0c1be42ae3f4540dfcb761bb.txt
extracting: export/labels/armas (295)_jpg_rf.fd3250a1d9bdedb4962e12465edcfaf21.txt
extracting: export/labels/armas (2482)_jpg_rf.rfd20edcfa50c86418ff964e954d021f.txt
extracting: export/labels/armas (22)_jpg_rf.feat5c3366f2d504e4848fcfb7efb348.txt
extracting: export/labels/armas (1970)_jpg_rf.fe0b97dd78fe9da3006aa60ab772e055.txt
extracting: export/labels/armas (1998)_jpg_rf.fec4e2d0789bf2f0023af292e75625e3.txt
extracting: export/labels/armas (2118)_jpg_rf.feaeebcb3b92a3887e1d2994212e084ef.txt
extracting: export/labels/armas (230)_jpg_rf.ff090ac29e9cd8cdde3d5823bb88d7f0c.txt
extracting: export/labels/armas (2490)_jpg_rf.ff618bce306ba6b618c050ff915d905.txt
extracting: export/labels/armas (2974)_jpg_rf.ff4e6c805d68a4fc746a7046859b0fab.txt
extracting: export/labels/armas (38)_jpg_rf.ff874c9502a4fd57c724b3479cbc5eba.txt
extracting: export/labels/armas (948)_jpg_rf.ff1208620e2107c5866c8276bea0d5c6.txt
extracting: export/labels/armas (339)_jpg_rf.ff862b5e10d0625cd44151c432315bcf.txt
extracting: export/labels/armas (1858)_jpg_rf.ff0769f74fe85518e04ce2e2b047f36d4.txt
extracting: export/labels/armas (389)_jpg_rf.ff7067813bf94de4f876536e66173c6b.txt
extracting: export/labels/armas (2735)_jpg_rf.ff5f9700c88320d9b1490fd860f28086.txt
extracting: export/labels/armas (1614)_jpg_rf.ffcca44ff2f8e51730498a8cfb30a0ac.txt
extracting: data.yaml
extracting: README.roboflow.txt
extracting: README.dataset.txt
```

ARMAS (1)_jpg_rf.c4150f819f9dc2ae75336e5b64e67d67.jpg



2023 DL PJK

Run YOLOv5 in Colab



Run YOLOv5 in Colab

Download YOLOv5 from github

Download env from requirements

The screenshot shows a Google Colab interface with two code cells and a file browser.

File Browser: On the left, a file browser shows the contents of the current directory, which includes a folder named "yolov5" and several Python files: Dockerfile, LICENSE, README.md, detect.py, hubconf.py, requirements.txt, test.py, train.py, and tutorial.ipynb.

Code Cell 1: This cell contains the command to clone the YOLOv5 repository from GitHub. A red arrow points from the "Download YOLOv5 from github" text box to this cell.

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

Code Cell 2: This cell contains the command to install the required Python packages listed in requirements.txt. A red arrow points from the "Download env from requirements" text box to this cell.

```
%cd /content/yolov5/  
!pip install -r requirements.txt
```

Bottom Status Bar: The status bar at the bottom indicates "76.74 GB 사용 가능" (76.74 GB available).

Run YOLOv5 in Colab

Check data.yaml file and edit

The screenshot shows a Jupyter Notebook interface in Google Colab. On the left, there's a file tree with 'dataset' and 'yolov5' folders. The main area displays terminal output:

```
Found existing installation: pycocotools 2.0.1
Uninstalling pycocotools-2.0.1:
Successfully uninstalled pycocotools-2.0.1
Successfully installed PyYAML-5.3.1 numpy-1.17.3 pycocotools-2.0
WARNING: The following packages were previously imported in this runtime:
[numpy]
You must restart the runtime in order to use newly installed versions.

RESTART RUNTIME
```

Below this, a code cell is shown:

```
[4] %cat /content/dataset/data.yaml
```

The output of this cell is:

```
train: ../train/images
val: ../valid/images

nc: 1
names: ['pistol']
```

A red arrow points from the text "Check data.yaml file and edit" to the code cell containing the command "%cat /content/dataset/data.yaml".

Run YOLOv5 in Colab

Check data.yaml file and edit

```
[7] with open('/content/dataset/train.txt', 'w') as f:  
    f.write('\n'.join(train_img_list) + '\n')  
  
with open('/content/dataset/val.txt', 'w') as f:  
    f.write('\n'.join(val_img_list) + '\n')  
  
import yaml  
  
with open('/content/dataset/data.yaml', 'r') as f:  
    data = yaml.load(f)  
  
    print(data)  
  
    data['train'] = '/content/dataset/train.txt'  
    data['val'] = '/content/dataset/val.txt'  
  
    with open('/content/dataset/data.yaml', 'w') as f:  
        yaml.dump(data, f)  
  
    print(data)
```

Run YOLOv5 in Colab

Read img data file and split data

```
val: ../valid/images
nc: 1
names: ['pistol']

[5] %cd /
from glob import glob

img_list = glob('/content/dataset/export/images/*.jpg')
print(len(img_list))

[6] /
2971

from sklearn.model_selection import train_test_split

train_img_list, val_img_list = train_test_split(img_list, test_size=0.2, random_state=2000)

print()
```

Run YOLOv5 in Colab

Read img data file and split data

```
val: ../valid/images
nc: 1
names: ['pistol']

[5] %cd /
from glob import glob

img_list = glob('/content/dataset/export/images/*.jpg')
print(len(img_list))

[6] / 2971

[6] from sklearn.model_selection import train_test_split

train_img_list, val_img_list = train_test_split(img_list, test_size=0.2, random_state=2000)

print(len(train_img_list), len(val_img_list))

[6] 2376 595
```



Run YOLOv5 in Colab

Create train, val data txt file

```
[6] from sklearn.model_selection import train_test_split

train_img_list, val_img_list = train_test_split(img_list, test_size=0.2, random_state=2000)

print(len(train_img_list), len(val_img_list))

[6]: 2376 595

[7]: with open('/content/dataset/train.txt', 'w') as f:
      f.write('\n'.join(train_img_list) + '\n')

[8]: with open('/content/dataset/val.txt', 'w') as f:
      f.write('\n'.join(val_img_list) + '\n')
```



Train

```
cd /content/yolov5/  
!python train.py --img 416 --batch 16 --epochs 50 --data /content/dataset/data.yaml --cfg ./models/yolov5s.yaml --weights yolov5s.pt --name gun_yolov5s_results
```

Run YOLOv5 in Colab

Train

```
%cd /content/yolov5/
```

```
!python train.py --img 416 --batch 16 --epochs 50 --data /content/dataset/data.yaml --cfg ./models/yolov5s.yaml --weights yolov5s.pt --name gun_yolov5s_results
```

```
/content/yolov5
Apex recommended for faster mixed precision training: https://github.com/NVIDIA/apex
...
Namespace(batch_size=16, bucket='', cache_images=False, cfg='./models/yolov5s.yaml', data='/content/dataset/data.yaml', device='Using CUDA device0 _CudaDeviceProperties(name='Tesla T4', total_memory=15079MB)

2020-07-15 03:55:39.065777: I tensorflow/stream_executor/platform/default/dso_loader.cc:44] Successfully opened dynamic library
Start Tensorboard with "tensorboard --logdir=runs", view at http://localhost:6006/
Hyperparameters {'optimizer': 'SGD', 'lr0': 0.01, 'momentum': 0.937, 'weight_decay': 0.0005, 'giou': 0.05, 'cls': 0.58, 'cls_p': Overriding ./models/yolov5s.yaml nc=80 with nc=1

      from    n      params    module                                arguments
 0           -1    1        3520  models.common.Focus                [3, 32, 3]
 1           -1    1       18560  models.common.Conv                 [32, 64, 3, 2]
 2           -1    1      19904  models.common.BottleneckCSP          [64, 64, 1]
 3           -1    1       73984  models.common.Conv                 [64, 128, 3, 2]
 4           -1    1      161152  models.common.BottleneckCSP          [128, 128, 3]
 5           -1    1       295424  models.common.Conv                [128, 256, 3, 2]
 6           -1    1      641792  models.common.BottleneckCSP          [256, 256, 3]
 7           -1    1     1180672  models.common.Conv                [256, 512, 3, 2]
 8           -1    1       656896  models.common.SPP                  [512, 512, [5, 9, 13]]
 9           -1    1      1248768  models.common.BottleneckCSP          [512, 512, 1, False]
10          -1    1       131584  models.common.Conv                [512, 256, 1, 1]
11          -1    1         0  torch.nn.modules.upsampling.Upsample  [None, 2, 'nearest']
12         [-1, 6]   1         0  models.common.Concat               [1]
13          -1    1      378624  models.common.BottleneckCSP          [512, 256, 1, False]
14          -1    1       33024  models.common.Conv                 [256, 128, 1, 1]
15          -1    1         0  torch.nn.modules.upsampling.Upsample  [None, 2, 'nearest']
16         [-1, 4]   1         0  models.common.Concat               [1]
17          -1    1      95104  models.common.BottleneckCSP          [256, 128, 1, False]
18          -1    1       2322  torch.nn.modules.conv.Conv2d          [128, 18, 1, 1]
19          -2    1      147712  models.common.Conv                 [128, 128, 3, 2]
20         [-1, 14]  1         0  models.common.Concat               [1]
21          -1    1      313088  models.common.BottleneckCSP          [256, 256, 1, False]
22          -1    1       4626  torch.nn.modules.conv.Conv2d          [256, 18, 1, 1]
23          -2    1      590336  models.common.Conv                 [256, 256, 3, 2]
24         [-1, 101] 1         0  models.common.Concat               [1]
```

Run YOLOv5 in Colab

Train

```
100 14.4M 0 14.4M 0 0 7862K 0 ----- 0:00:01 ----- 261M
Downloading https://drive.google.com/uc?export=download&id=1R5T6rIyy3lIwgFXNms8whc-387H0tMQQ as yolov5s.pt... Done (4.8s)
Scanning images: 100% 2376/2376 [00:00<00:00, 7681.61it/s]
Scanning labels /content/dataset/export/labels.cache (2376 found, 0 missing, 0 empty, 0 duplicate, for 2376 images): 100% 2376
Scanning images: 100% 595/595 [00:00<00:00, 8824.77it/s]
Scanning labels /content/dataset/export/labels.cache (595 found, 0 missing, 0 empty, 0 duplicate, for 595 images): 100% 595

Analyzing anchors... Best Possible Recall (BPR) = 1.0000
Image sizes 416 train, 416 test
Using 2 dataloader workers
Starting training for 50 epochs...

Epoch    gpu_mem      GIoU      obj      cls      total      targets      img_size
  0/49    2.18G    0.0871    0.06698      0    0.1541      15      416: 100% 149/149 [00:40<00:00, 3.63it/s]
          Class      Images      Targets      P          R      mAP@.5  mAP@.5:.95: 100% 38/38 [00:06<00:00, 6.13it/s]
          all        595         702      0.173      0.821      0.629      0.348
          Epoch    gpu_mem      GIoU      obj      cls      total      targets      img_size
  1/49    2.18G    0.05653   0.06028      0    0.1168      20      416: 100% 149/149 [00:38<00:00, 3.90it/s]
          Class      Images      Targets      P          R      mAP@.5  mAP@.5:.95: 100% 38/38 [00:04<00:00, 9.16it/s]
          all        595         702      0.348      0.806      0.589      0.329
          Epoch    gpu_mem      GIoU      obj      cls      total      targets      img_size
  2/49    2.18G    0.05033   0.0584       0    0.1087      33      416: 1% 2/149 [00:00<01:16, 1.91it/s]
```

```
[ ]
```

Run YOLOv5 in Colab

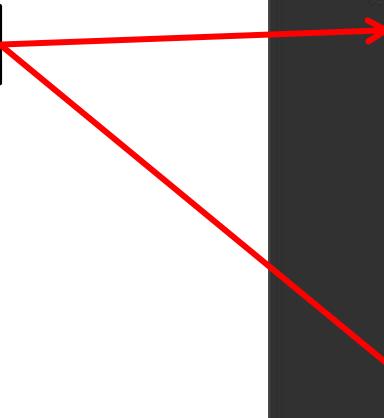
Train

```
          Class      Images     Targets       P       R   mAP@.5  mAP@.5:.95: 100% 38/38 [00:03<00:00,  9.81i
          all       595        702      0.752      0.903      0.914      0.654
Epoch  gpu_mem      GIoU      obj      cls      total    targets  img_size
47/49   2.18G   0.025   0.03918      0  0.06419      18      416: 100% 149/149 [00:37<00:00,  4.00it/s]
          Class      Images     Targets       P       R   mAP@.5  mAP@.5:.95: 100% 38/38 [00:03<00:00,  9.72i
          all       595        702      0.707      0.895      0.904      0.645
Epoch  gpu_mem      GIoU      obj      cls      total    targets  img_size
48/49   2.18G   0.02473   0.03874      0  0.06347      19      416: 100% 149/149 [00:37<00:00,  3.98it/s]
          Class      Images     Targets       P       R   mAP@.5  mAP@.5:.95: 100% 38/38 [00:03<00:00,  9.75i
          all       595        702      0.735      0.886      0.905      0.646
Epoch  gpu_mem      GIoU      obj      cls      total    targets  img_size
49/49   2.18G   0.02471   0.03728      0  0.06199      16      416: 100% 149/149 [00:37<00:00,  3.93it/s]
          Class      Images     Targets       P       R   mAP@.5  mAP@.5:.95: 100% 38/38 [00:04<00:00,  9.48i
          all       595        702      0.727      0.89      0.904      0.649
Optimizer stripped from runs/exp0_gun_yolov5s_results/weights/last_gun_yolov5s_results.pt, 14.7MB
Optimizer stripped from runs/exp0_gun_yolov5s_results/weights/best_gun_yolov5s_results.pt, 14.7MB
50 epochs completed in 0.583 hours.
```

Run YOLOv5 in Colab

Train

Check output



```
var.txt
sample_data
yolov5
  data
  inference
  models
  runs
    exp0_gun_yolov5s_results
      1594785364.8637826
      weights
      events.out.tfevents.15947...
      hyp.yaml
      labels.png
      opt.yaml
      results.txt
      test_batch0_gt.jpg
      test_batch0_pred.jpg
      train_batch0.jpg
      train_batch1.jpg
      train_batch2.jpg
      utils
      weights
      Dockerfile
      LICENSE
      README.md
      detect.py
      hubconf.py
```

```
Scanning images: 100% 2376/2376 [00:00<00:00, 7681.61it/s]
Scanning labels /content/dataset/export/labels.cache (2376 found, 0 missing, 0 empty)
Scanning images: 100% 595/595 [00:00<00:00, 8824.77it/s]
Scanning labels /content/dataset/export/labels.cache (595 found, 0 missing, 0 empty)

Analyzing anchors... Best Possible Recall (BPR) = 1.0000
Image sizes 416 train, 416 test
Using 2 dataloader workers
Starting training for 50 epochs...

Epoch 0/49   gpu_mem     GIoU     obj     cls   total   targets   img_size
          2.18G   0.0871   0.06698    0   0.1541      15   416: 10
          Class   Images   Targets   P       R   mAP@.5   mA
          all      595      702      0.173   0.821   0.629
Epoch 1/49   gpu_mem     GIoU     obj     cls   total   targets   img_size
          2.18G   0.05653   0.06028    0   0.1168      20   416: 10
          Class   Images   Targets   P       R   mAP@.5   mA
          all      595      702      0.348   0.806   0.589
Epoch 2/49   gpu_mem     GIoU     obj     cls   total   targets   img_size
          2.18G   0.04884   0.05788    0   0.1067      15   416: 10
          Class   Images   Targets   P       R   mAP@.5   mA
          all      595      702      0.407   0.829   0.766
Epoch 3/49   gpu_mem     GIoU     obj     cls   total   targets   img_size
          2.18G   0.04823   0.05844    0   0.1067      36   416: 5
```

Run YOLOv5 in Colab

Train

Check output

The screenshot shows a Jupyter Notebook interface with two main sections: a file browser on the left and a terminal output on the right.

File Browser:

- Var.txt
- sample_data
- yolov5 (selected)
- data
- inference
- models
- runs
- exp0_gun_yolov5s_results (selected)
- 1594785364.8637826
- weights
- events.out.tfevents.15947...
- hyp.yaml
- labels.png
- opt.yaml
- results.txt
- test_batch0_gt.jpg
- test_batch0_pred.jpg** (highlighted with a red arrow)
- train_batch0.jpg
- train_batch1.jpg
- train_batch2.jpg
- utils
- weights
- Dockerfile
- LICENSE
- README.md
- detect.py
- hubconf.py

Terminal Output:

```
Scanning images: 100% 2376/2376 [00:00<00:00, 7681.61it/s]
Scanning labels /content/dataset/export/labels.cache (2376 found, 0 missing, 0 empty)
Scanning images: 100% 595/595 [00:00<00:00, 8824.77it/s]
Scanning labels /content/dataset/export/labels.cache (595 found, 0 missing, 0 empty)

Analyzing anchors... Best Possible Recall (BPR) = 1.0000
Image sizes 416 train, 416 test
Using 2 dataloader workers
Starting training for 50 epochs...

Epoch 0/49  gpu_mem    GIoU      obj      cls      total      targets      img_size
          2.18G  0.0871  0.06698      0  0.1541      15      416: 10
          Class   Images     Targets      P      R  mAP@.5  mA
          all      595       702      0.173      0.821      0.629
Epoch 1/49  gpu_mem    GIoU      obj      cls      total      targets      img_size
          2.18G  0.05653  0.06028      0  0.1168      20      416: 10
          Class   Images     Targets      P      R  mAP@.5  mA
          all      595       702      0.348      0.806      0.589
Epoch 2/49  gpu_mem    GIoU      obj      cls      total      targets      img_size
          2.18G  0.04884  0.05788      0  0.1067      15      416: 10
          Class   Images     Targets      P      R  mAP@.5  mA
          all      595       702      0.407      0.829      0.766
Epoch 3/49  gpu_mem    GIoU      obj      cls      total      targets      img_size
          2.18G  0.04823  0.05844      0  0.1067      36      416: 5
```

Run YOLOv5 in Colab

Train

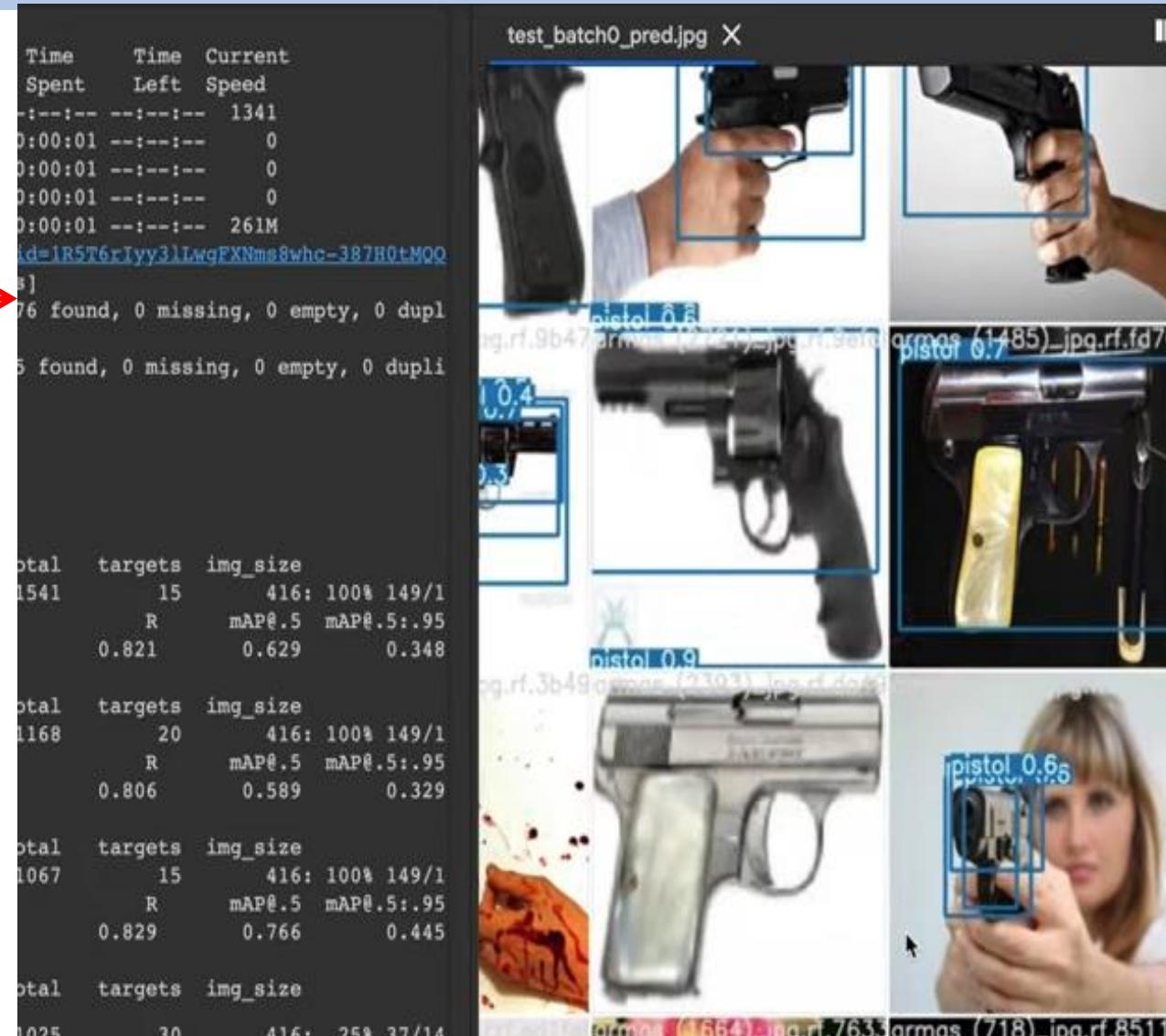
Check output



Run YOLOv5 in Colab

Train

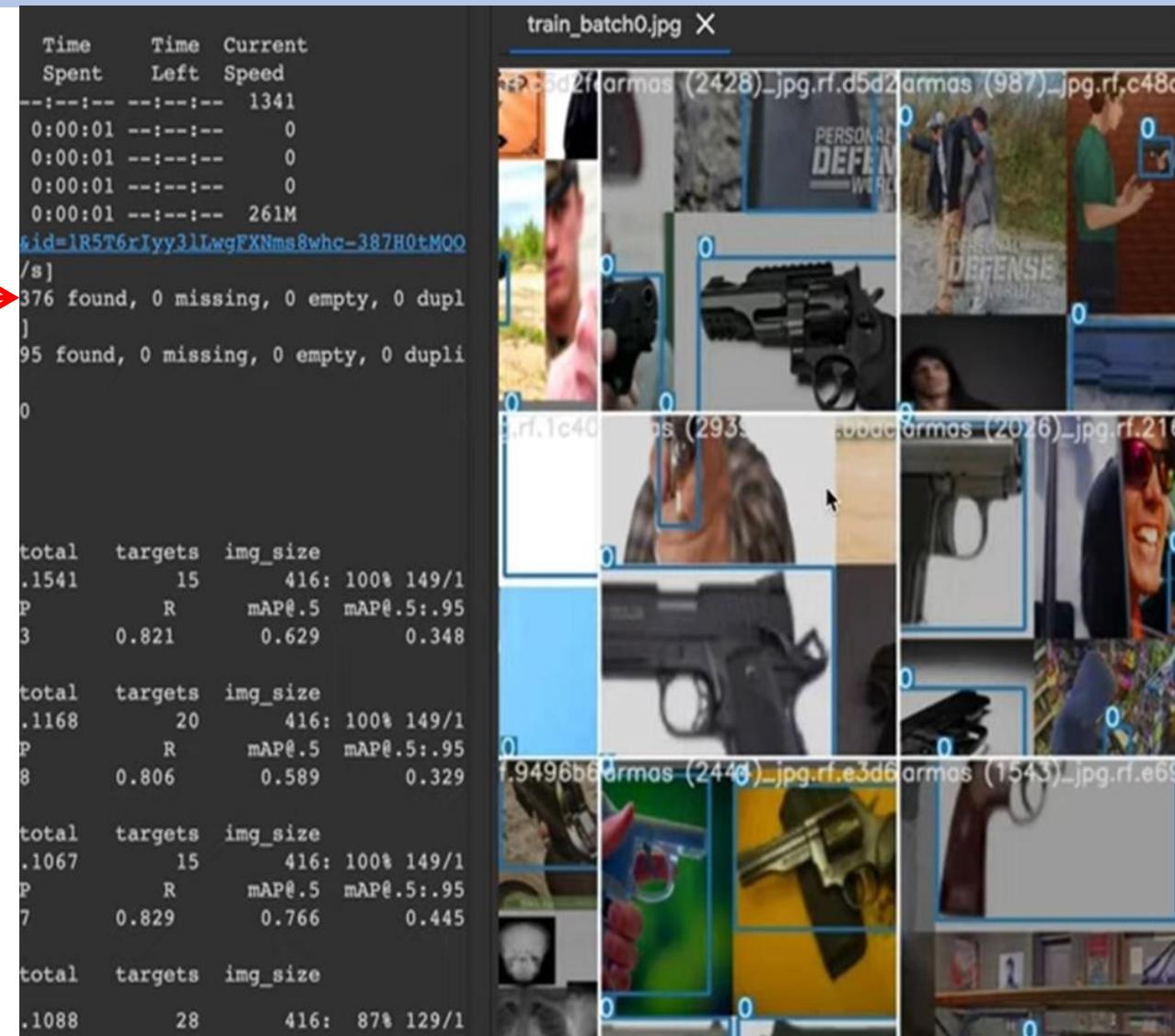
Check output



Run YOLOv5 in Colab

Train

Check output



Run YOLOv5 in Colab

Train

Check output

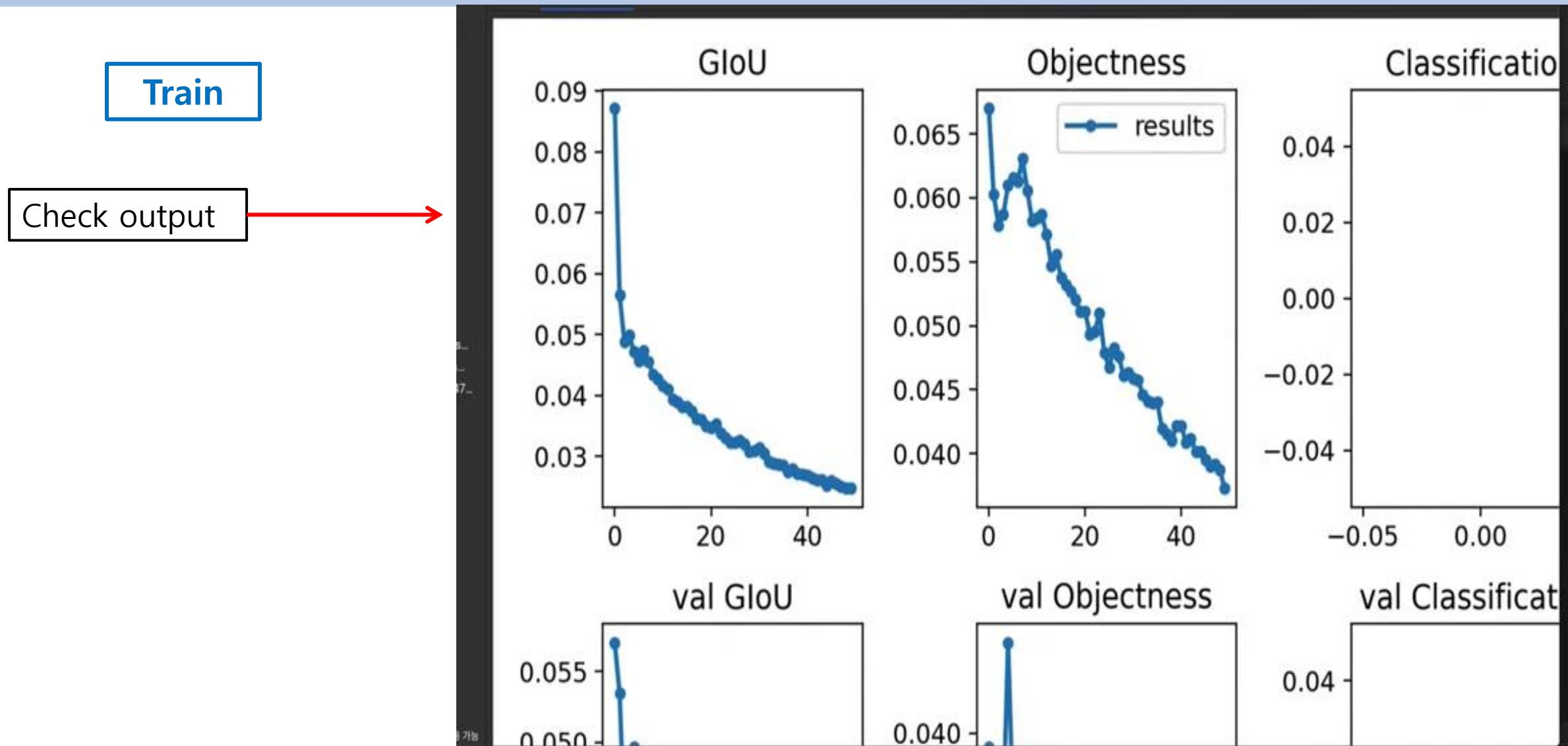
The screenshot shows a Google Colab interface with two main sections: a file browser on the left and a terminal window on the right.

File Browser: The left pane displays the directory structure of the project. It includes files like `README.roboflow.txt`, `data.yaml`, `train.txt`, `val.txt`, and several folders: `sample_data`, `yolov5`, `data`, `inference`, `models`, and `runs`. The `runs` folder contains sub-folders for experiments, with one highlighted as `exp0_gun_yolov5s_results`. Inside this experiment folder, there are sub-folders for `1594785364.8637826` and `weights`. The `weights` folder contains files named `best_gun_yolov5s_res...` and `last_gun_yolov5s_res...`.

Terminal Window: The right pane shows the training logs for YOLOv5. The logs are organized by epoch, from 44 to 49. Each epoch log includes metrics such as GPU memory usage (`gpu_mem`), GIoU, object detection rate (`obj`), classification rate (`cls`), total error, target count, and image size. The logs also mention the number of images (`Images`) and targets (`Targets`) processed. The final message indicates that 50 epochs were completed in 0.583 hours.

```
Epoch 44/49 gpu_mem 2.18G GIoU 0.02522 obj 0.04017 cls 0 total 0.06538 targets 15 img_size 416: 10
Class Images Targets P R mAP@.5 mA
all 595 702 0.716 0.902 0.913
Epoch 45/49 gpu_mem 2.18G GIoU 0.02593 obj 0.03954 cls 0 total 0.06547 targets 19 img_size 416: 10
Class Images Targets P R mAP@.5 mA
all 595 702 0.712 0.905 0.914
Epoch 46/49 gpu_mem 2.18G GIoU 0.02552 obj 0.03905 cls 0 total 0.06457 targets 20 img_size 416: 10
Class Images Targets P R mAP@.5 mA
all 595 702 0.752 0.903 0.914
Epoch 47/49 gpu_mem 2.18G GIoU 0.025 0.03918 cls 0 total 0.06419 targets 18 img_size 416: 10
Class Images Targets P R mAP@.5 mA
all 595 702 0.707 0.895 0.904
Epoch 48/49 gpu_mem 2.18G GIoU 0.02473 0.03874 cls 0 total 0.06347 targets 19 img_size 416: 10
Class Images Targets P R mAP@.5 mA
all 595 702 0.735 0.886 0.905
Epoch 49/49 gpu_mem 2.18G GIoU 0.02471 0.03728 cls 0 total 0.06199 targets 16 img_size 416: 10
Class Images Targets P R mAP@.5 mA
all 595 702 0.727 0.89 0.904
Optimizer stripped from runs/exp0_gun_yolov5s_results/weights/last_gun_yolov5s_res...
Optimizer stripped from runs/exp0_gun_yolov5s_results/weights/best_gun_yolov5s_res...
50 epochs completed in 0.583 hours.
```

Run YOLOv5 in Colab



Run YOLOv5 in Colab

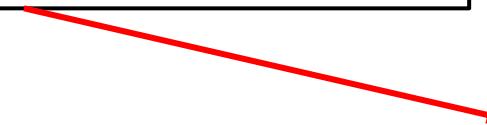
Train

Check output



Run YOLOv5 in Colab

Use tensorboard for checking the data

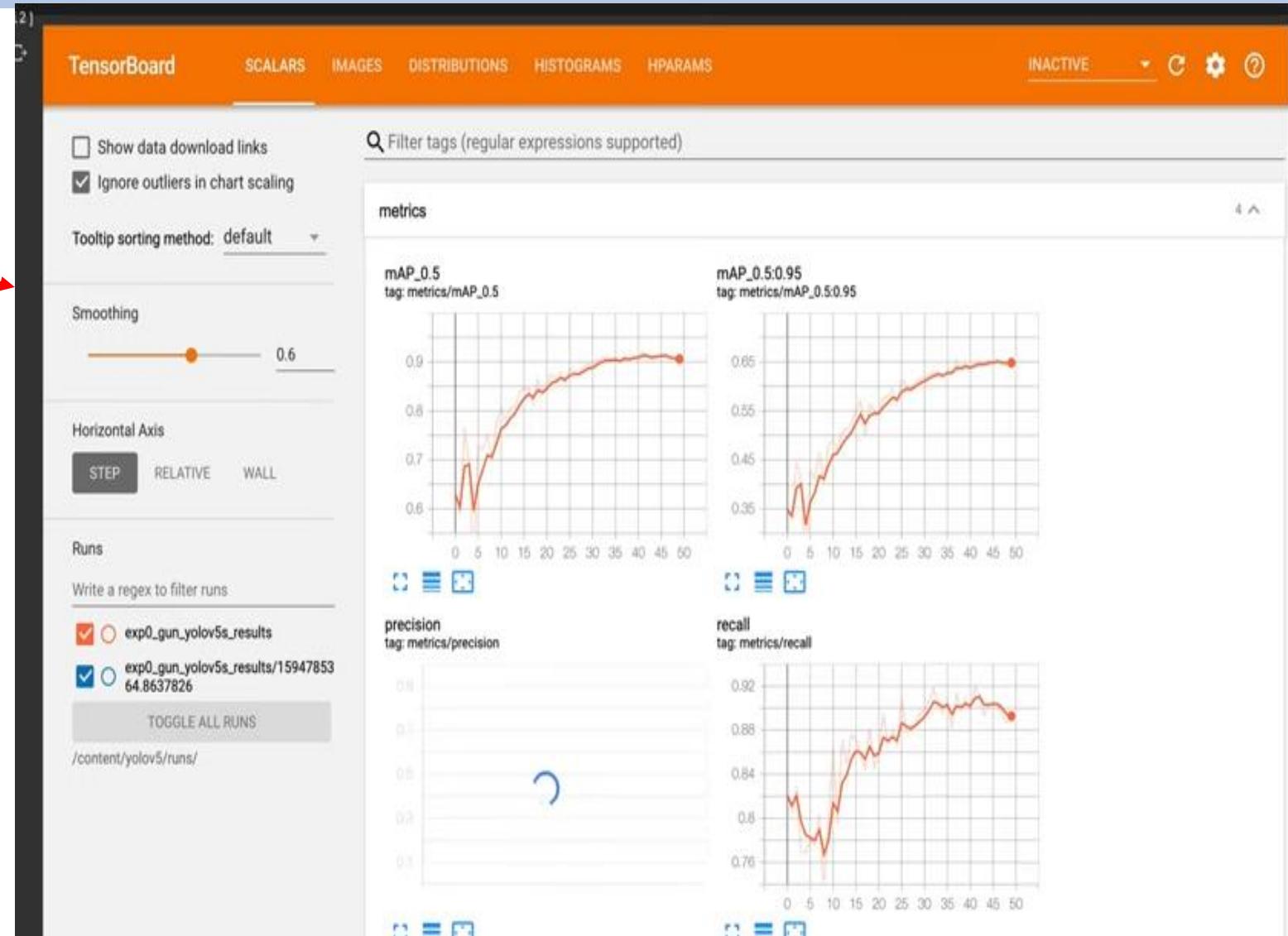


```
Epoch    gpu_mem      GIoU      obj      cls      total      targets      img_size
49/49    2.18G    0.02471    0.03728      0    0.06199      16      416: 100% 149/149 [00:37<00:00,  3.93it/s]
          Class      Images      Targets      P      R      mAP@.5: mAP@.5:.95: 100% 38/38 [00:04<00:00,  9.48i
          all       595       702      0.727      0.89      0.904      0.649
Optimizer stripped from runs/exp0_gun_yolov5s_results/weights/last_gun_yolov5s_results.pt, 14.7MB
Optimizer stripped from runs/exp0_gun_yolov5s_results/weights/best_gun_yolov5s_results.pt, 14.7MB
50 epochs completed in 0.583 hours.

%load_ext tensorboard
!tensorboard --logdir /content/yolov5/runs/
... Launching TensorBoard...
```

Run YOLOv5 in Colab

Use tensorboard for checking the data



Done !!!

➤ Make your own **YOLO** model !

- ✓ Any YOLO
- ✓ Due to **1000am, May 9 2023 (before the class)**
- ✓ **Upload as PDF file and video clip on LMS only** (don't need hardcopy)

You can't learn anything if you don't do it yourself !!!