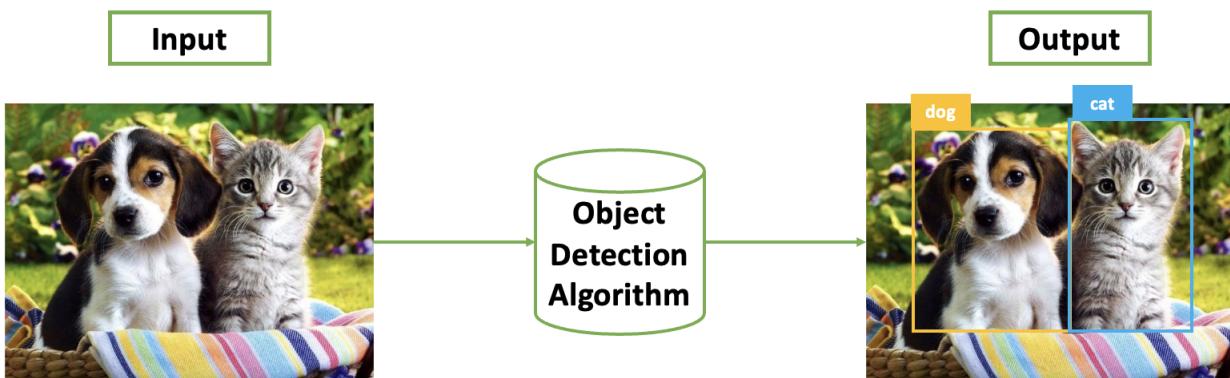


Object Detection with YOLOv8 – Project

Ngày 20 tháng 5 năm 2023

Phần I: Giới thiệu

Phát hiện đối tượng (Object Detection) là một bài toán thuộc lĩnh vực Thị giác máy tính (Computer Vision), trong đó nhiệm vụ của chúng ta là xây dựng một chương trình có thể trả về các tọa độ (bounding box) và tên phân lớp (class name) của những vật thể có trong ảnh mà ta mong muốn tìm kiếm (detection).



Hình 1: Input/Output của bài toán Object Detection với vật thể cần xác định là Chó, Mèo.



Hình 2: Ví dụ minh họa kết quả của chương trình Human Detection sử dụng YOLOv8.

Trong project này, chúng ta sẽ tìm hiểu và thực hành cài đặt mã nguồn chương trình Python về Phát hiện con người (Human Detection) có trong một tấm ảnh sử dụng thuật toán có tên gọi là **YOLOv8**.

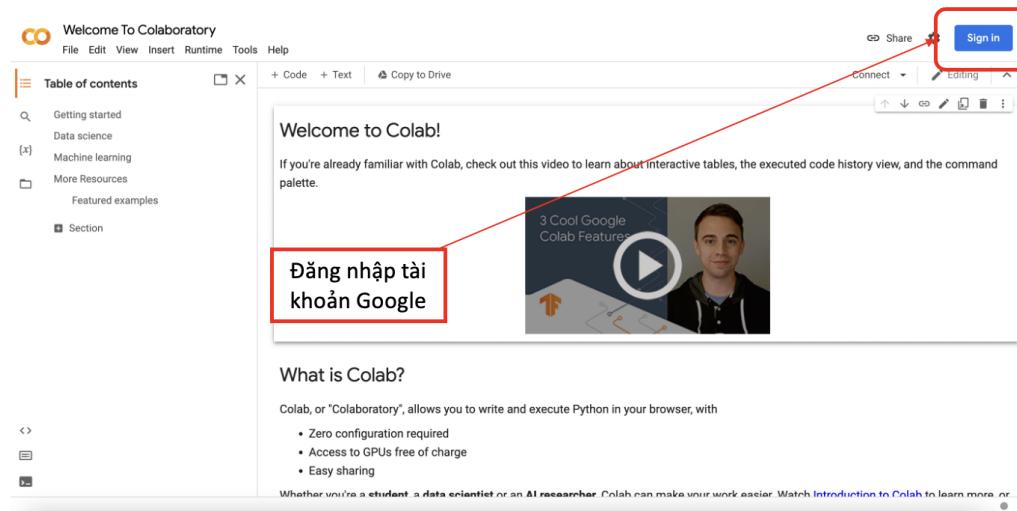
Yêu cầu project: Các bạn hãy đọc, tìm hiểu cách sử dụng mã nguồn YOLOv8 theo hướng dẫn bên dưới và áp dụng vào bộ dữ liệu **human**. Sau đó hoàn thành phần câu hỏi trắc nghiệm.

Phần II: Cài đặt chương trình

Để cài đặt và sử dụng thuật toán YOLOv8, các bạn hãy thực hiện theo các bước hướng dẫn sau:

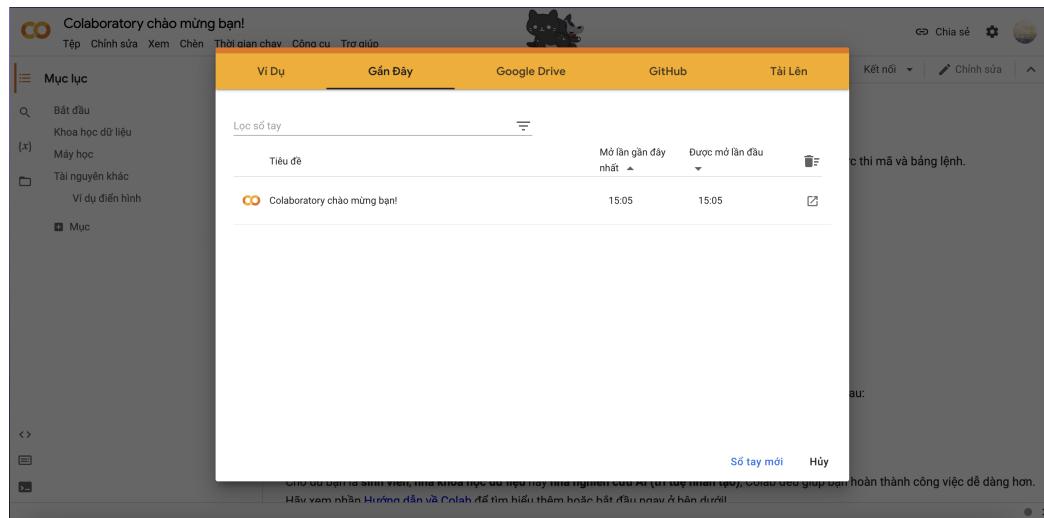
1. **Chuẩn bị môi trường cài đặt:** Để thuận tiện trong việc sử dụng YOLOv8, chúng ta sẽ dùng Google Colab làm môi trường cài đặt. Các bước sử dụng Google Colab được thực hiện như sau:

- **Bước 1:** Truy cập vào đường dẫn sau: [link](#). Nếu truy cập thành công, các bạn sẽ thấy giao diện như hình dưới đây:



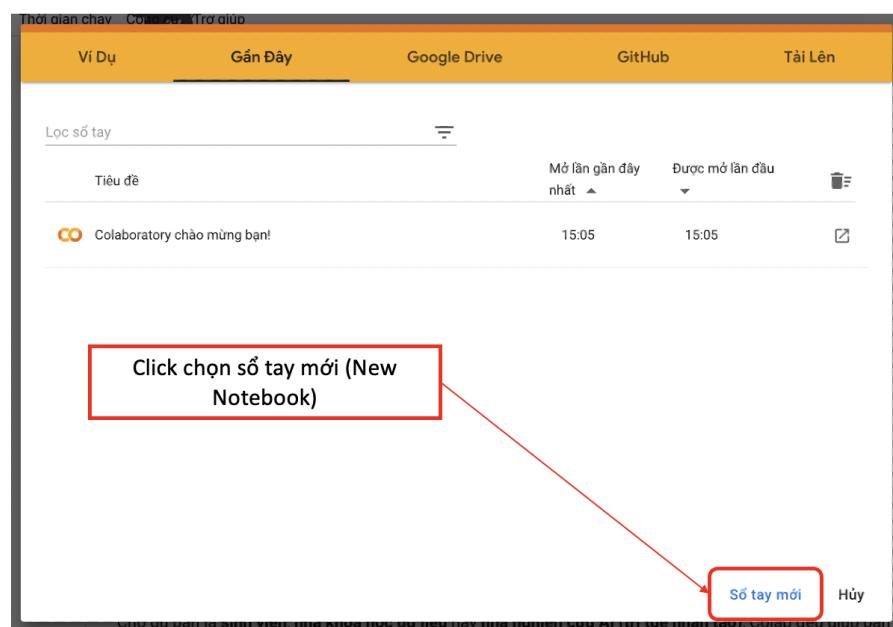
Hình 3: Giao diện chính của Google Colab

Sau đó, các bạn hãy đăng nhập bằng tài khoản Google của mình. Nếu đăng nhập thành công, một cửa sổ mới sẽ hiện lên như hình dưới đây:



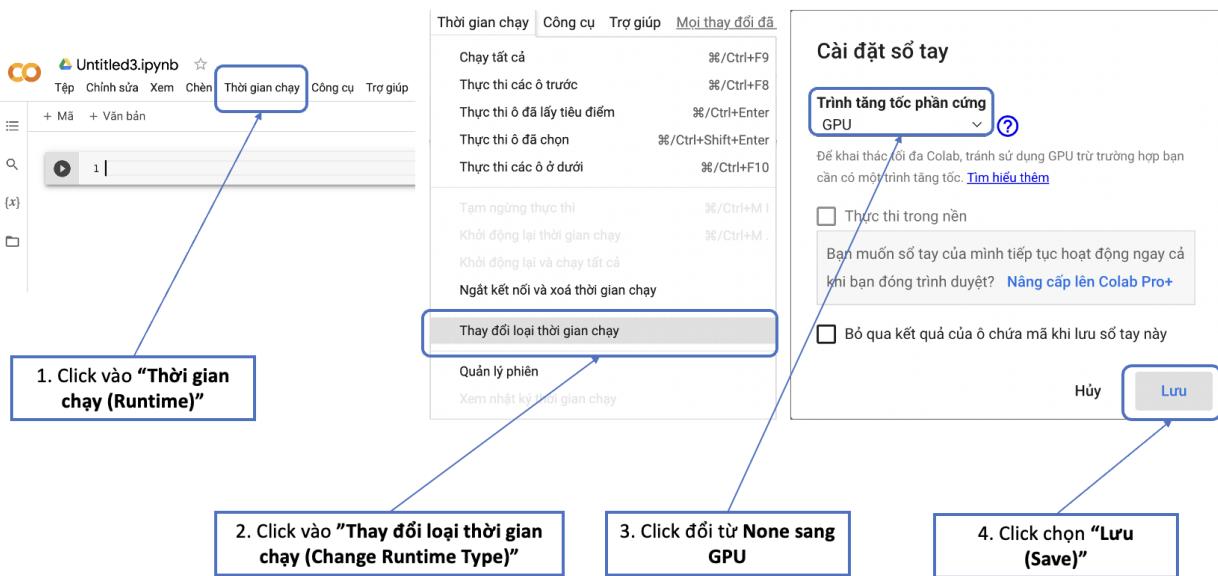
Hình 4: Giao diện Google Colab sau khi đăng nhập thành công

- **Bước 2:** Khởi tạo notebook mới. Notebook sẽ là giao diện để ta có thể viết các dòng lệnh Python. Để tạo được notebook, các bạn thực hiện thao tác theo hình dưới đây:



Hình 5: Khởi tạo notebook mới

- **Bước 3:** Thay đổi runtime của notebook từ CPU thành GPU.



Hình 6: Các bước kích hoạt GPU cho notebook mới trên Google Colab

- **Bước 4:** Cuối cùng, để có thể thực thi các dòng lệnh Python, ta cần khởi động notebook. Các thao tác khởi động một notebook trong Google Colab sẽ như hình sau:

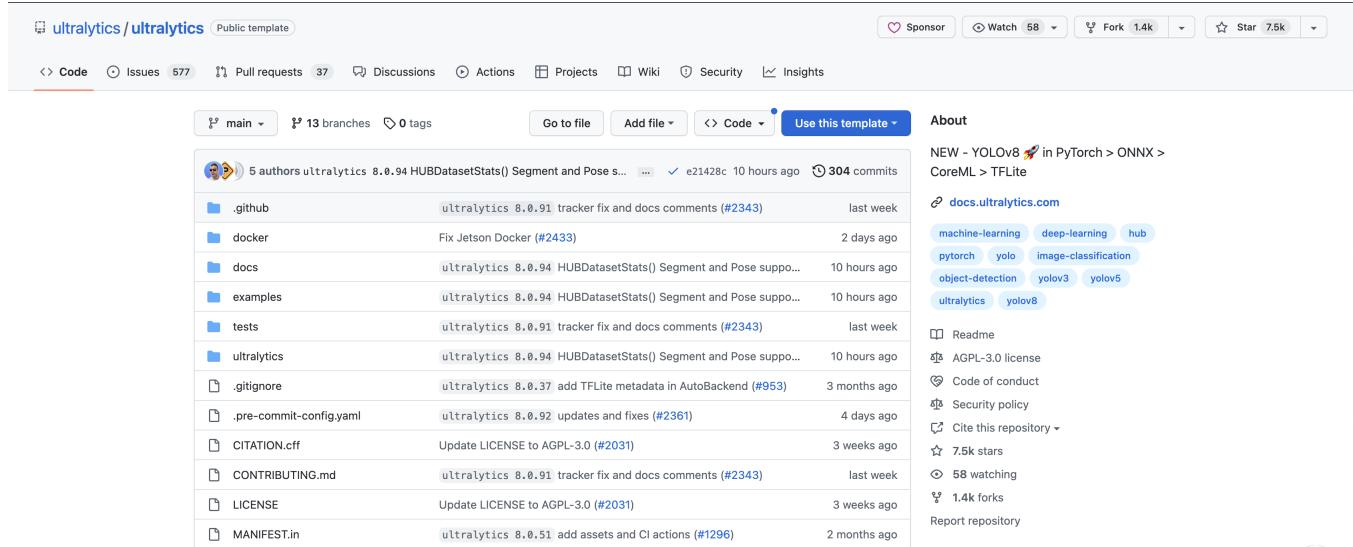


Hình 7: Khởi động một notebook có GPU trong Google Colab

Sau khi thực hiện các bước trên, các bạn đã có một môi trường code Python với GPU miễn phí từ Google.

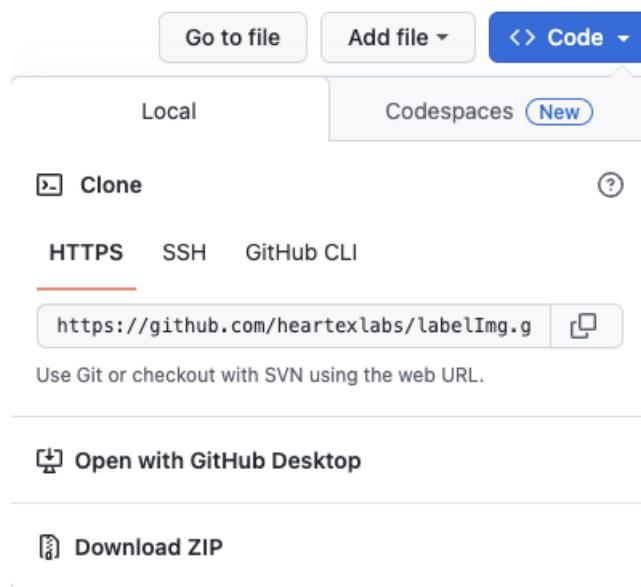
2. **Tải mã nguồn YOLOv8 từ GitHub:** Để sử dụng YOLOv8, chúng ta cần tải mã nguồn (source code) của YOLOv8 về môi trường cài đặt code, mã nguồn của YOLOv8 được công khai trên GitHub. Như vậy, các bạn sẽ thực hiện theo các bước sau:

- **Bước 1:** Các bạn truy cập vào đường dẫn GitHub của YOLOv8 tại [đây](#). Nếu truy cập thành công, các bạn sẽ thấy giao diện như hình dưới:



Hình 8: Giao diện GitHub của YOLOv8

- **Bước 2:** Tại trang GitHub của YOLOv8, các bạn chọn mục **Code** (có màu nền xanh dương như ảnh dưới) và chọn nút copy đường dẫn như ảnh minh họa dưới đây:



Hình 9: Copy đường dẫn để tải mã nguồn YOLOv8

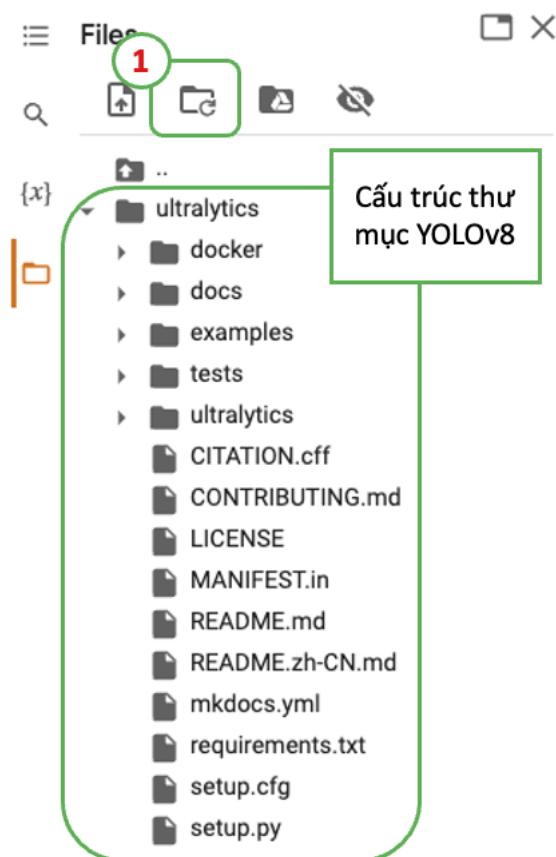
- **Bước 3:** Quay lại Google Colab, các bạn khởi tạo một code cell sử dụng lệnh: `!git clone <link>` (trong đó `<link>` là đường dẫn GitHub đã copy ở **bước 2**). Nếu code cell thực thi thành công, ta kết quả sẽ hiển thị như hình sau:

```
1 !git clone https://github.com/ultralytics/ultralytics

Cloning into 'ultralytics'...
remote: Enumerating objects: 8311, done.
remote: Counting objects: 100% (424/424), done.
remote: Compressing objects: 100% (235/235), done.
remote: Total 8311 (delta 264), reused 322 (delta 189), pack-reused 7887
Receiving objects: 100% (8311/8311), 5.91 MiB | 18.56 MiB/s, done.
Resolving deltas: 100% (5595/5595), done.
```

Hình 10: Tải mã nguồn YOLOv8 sử dụng lệnh git clone

Để kiểm tra, các bạn có thể refresh lại phần **Files** của Google Colab để xem thư mục YOLOv8 đã xuất hiện hay chưa.



Hình 11: Thư mục YOLOv8

3. Cài đặt các gói thư viện Python cần thiết theo yêu cầu của YOLOv8: Mã nguồn YOLOv8 được xây dựng bằng rất nhiều các thư viện Python khác nhau. Vì vậy, để có thể chạy được YOLOv8 ta cần tải các gói thư viện cần thiết mà YOLOv8 yêu cầu. YOLOv8 có hỗ trợ sẵn thư viện tên **ultralytics**, chúng ta có thể tải thư viện này thông qua lệnh pip như sau:

```

1 %cd ultralytics
2 !pip install ultralytics
3 import ultralytics
4
5 ultralytics.checks()

Ultralytics YOLOv8.0.104 🚀 Python-3.10.11 torch-2.0.0+cu118 CUDA:0 (Tesla T4, 15102MiB)
Setup complete ✅ (2 CPUs, 12.7 GB RAM, 23.4/78.2 GB disk)

```

Hình 12: Tải thư viện ultralytics dùng lệnh pip

Ngoài cách trên, ta cũng có thể cài đặt các gói thư viện bằng cách sử dụng file setup có sẵn trong mã nguồn của YOLOv8, các bạn có thể làm như sau:

```

1 %cd ultralytics
2 !pip install -e .

/content/ultralytics
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Obtaining file:///content/ultralytics
  Preparing metadata (setup.py) ... done
Requirement already satisfied: matplotlib>=3.2.2 in /usr/local/lib/python3.10/dist-packages (from ultralytics==8.0.104) (3.7.1)
Requirement already satisfied: opencv-python>=4.6.0 in /usr/local/lib/python3.10/dist-packages (from ultralytics==8.0.104) (4.7.0.72)
Requirement already satisfied: Pillow>=7.1.2 in /usr/local/lib/python3.10/dist-packages (from ultralytics==8.0.104) (8.4.0)
Requirement already satisfied: PyYAML>=5.3.1 in /usr/local/lib/python3.10/dist-packages (from ultralytics==8.0.104) (6.0)
Requirement already satisfied: requests>=2.23.0 in /usr/local/lib/python3.10/dist-packages (from ultralytics==8.0.104) (2.27.1)
Requirement already satisfied: scipy>=1.4.1 in /usr/local/lib/python3.10/dist-packages (from ultralytics==8.0.104) (1.10.1)
Requirement already satisfied: torch>=1.7.0 in /usr/local/lib/python3.10/dist-packages (from ultralytics==8.0.104) (2.0.0+cu118)
Requirement already satisfied: torchvision>=0.8.1 in /usr/local/lib/python3.10/dist-packages (from ultralytics==8.0.104) (0.15.1+cu118)

```

Hình 13: Tải các gói cài đặt cần thiết thông qua file setup

- Tải Pretrained Model:** Các bạn tải file pretrained model tại [đây](#) và đặt file đã tải vào thư mục `./ultralytics`. Trên Google Colab, việc này có thể được thực hiện thông qua lệnh `wget` như hình dưới đây:

```

1 !wget https://github.com/ultralytics/assets/releases/download/v0.0.0/yolov8s.pt

--2023-05-17 10:14:55-- https://github.com/ultralytics/assets/releases/download/v0.0.0/yolov8s.pt
Resolving github.com (github.com)... 192.30.255.112
Connecting to github.com (github.com)|192.30.255.112|:443... connected.
HTTP request sent, awaiting response... 302 Found
Location: https://objects.githubusercontent.com/github-production-release-asset-2e65be/521807533/404b29b7-e374
--2023-05-17 10:14:56-- https://objects.githubusercontent.com/github-production-release-asset-2e65be/521807533/404b29b7-e374
Resolving objects.githubusercontent.com (objects.githubusercontent.com)... 185.199.108.133, 185.199.109.133, 1...
Connecting to objects.githubusercontent.com (objects.githubusercontent.com)|185.199.108.133|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 22573363 (22M) [application/octet-stream]
Saving to: 'yolov8s.pt'

yolov8s.pt          100%[=====] 21.53M 70.9MB/s    in 0.3s

2023-05-17 10:14:56 (70.9 MB/s) - 'yolov8s.pt' saved [22573363/22573363]

```

Hình 14: Tải pretrained model sử dụng lệnh wget

Để kiểm tra xem file pretrained model đã được tải về thành công hay chưa, các bạn refresh phần **Files** của Google Colab và tìm kiếm file có tên **YOLOv8s.pt**:



Hình 15: File pretrained model

5. Chuẩn bị dữ liệu: Để huấn luyện YOLOv8 trên một bộ dữ liệu bất kì, chúng ta cần hai thành phần chính như sau:

- **Thư mục chứa dữ liệu:** Các bạn cần chuẩn bị một thư mục chứa bộ dữ liệu (thư mục này nằm trong thư mục **YOLOv8**) có cấu trúc cây thư mục như sau:



Hình 16: Tổ chức thư mục bộ dữ liệu trong YOLOv8

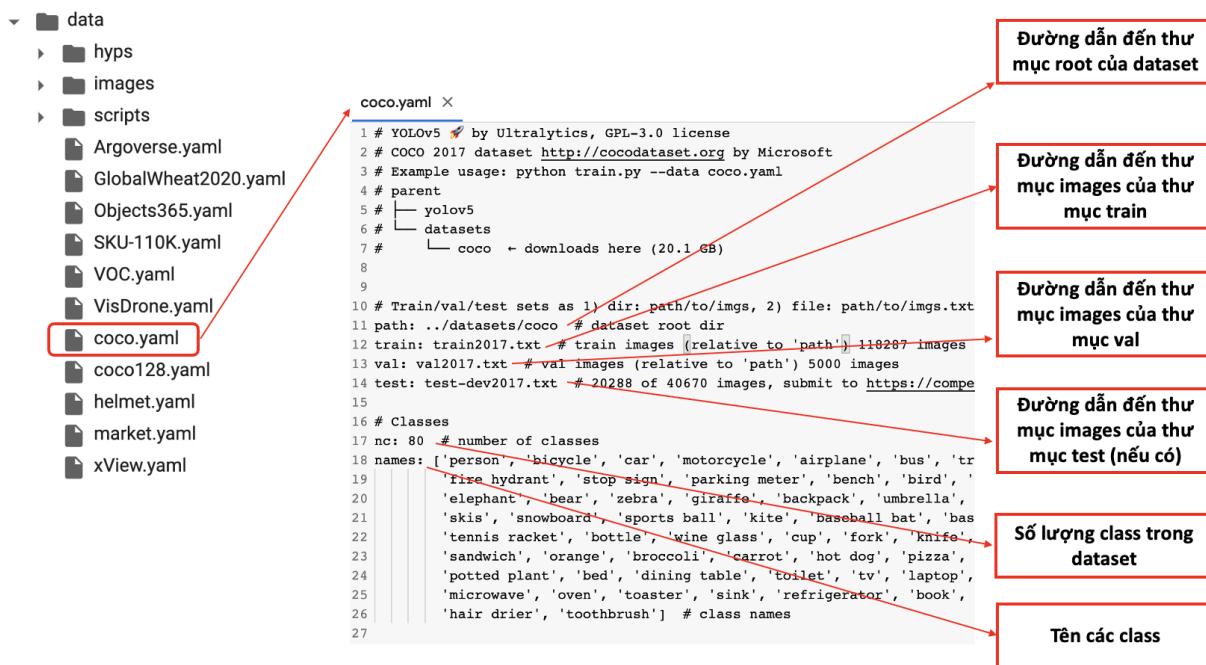
Trong đó:

- **Thư mục images:** Chứa các file ảnh (.jpg, .png...).
- **Thư mục labels:** Chứa các file .txt tương ứng với từng ảnh (có cùng tên file trong thư mục images).
- **File data.yaml:** Nội dung chi tiết ở phần sau.

Bộ dữ liệu human các bạn tải tại [đây](#). Các bạn có thể tải thủ công file .zip này về hoặc sử dụng các dòng lệnh sau để tải và giải nén file tự động:

```
1 !gdown https://drive.google.com/u/0/uc?id=1--0QuKMwj31K-CSvD8oq5fceFweifPPuN&
    export=download
2 !unzip /content/human_detection_dataset.zip
```

- **File .yaml:** Các bạn cần chuẩn bị một file .yaml chứa các thông tin về bộ dữ liệu phía trên trong thư mục **data**, ta có thể coi một số file .yaml mẫu được tác giả YOLOv8 cung cấp sẵn:



Hình 17: Các trường thông tin quan trọng trong file .yaml

Như vậy, đối với bộ dữ liệu **human**, ta chỉ việc tạo một file .yaml mới (ví dụ data.yaml) và điền các trường thông tin tương ứng (thư mục dữ liệu đã cung cấp sẵn file này cho các bạn). Ở đây các bạn có thể thực hiện thủ công hoặc sử dụng các dòng lệnh bên dưới đây để tạo tự động. Lưu ý rằng file .yaml cần được đặt ở trong thư mục của bộ dữ liệu **human_detection_dataset**:

```
1 import yaml
2
3 dataset_info = {
4     'train': './train/images',
5     'val': './val/images',
6     'nc': 1,
7     'names': ['Human']
8 }
9
10 with open('./human_detection_dataset/data.yaml', 'w+') as f:
11     doc = yaml.dump(dataset_info, f, default_flow_style=None, sort_keys=False)
```

Mỗi bộ dữ liệu khác nhau sẽ có nội dung thông tin khác nhau, vì vậy các bạn cần sẽ có những điều chỉnh cho phù hợp với bộ dữ liệu mà mình sử dụng (số lượng class, tên các class, đường dẫn đến thư mục dữ liệu...).

6. Thực hiện huấn luyện: Sau khi đã hoàn tất các bước chuẩn bị trên, chúng ta sẽ thực hiện huấn luyện (training) với bộ dữ liệu đã chuẩn bị, các bạn hãy thực thi dòng lệnh dưới đây (đối với dữ liệu khác các bạn cần thay đổi tên file .yaml tương ứng):

```
1 !yolo train model=yolov8s.pt data=./human_detection_dataset/data.yaml epochs=20
    imgsz=640
```

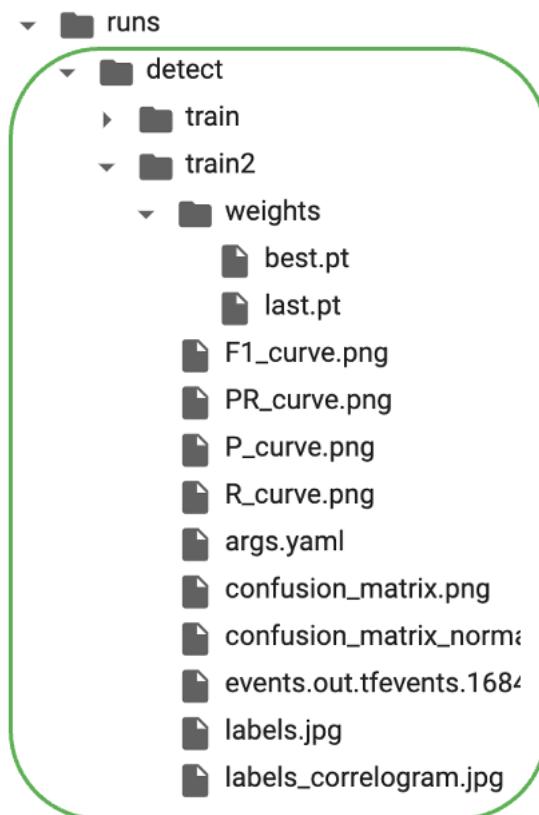
Sau khi kết thúc quá trình training, các bạn sẽ thấy nội dung in màn hình có dạng như hình minh họa dưới đây:

```
20 epochs completed in 2.010 hours.
Optimizer stripped from /content/ultralytics/runs/detect/train2/weights/last.pt, 22.5MB
Optimizer stripped from /content/ultralytics/runs/detect/train2/weights/best.pt, 22.5MB

Validating /content/ultralytics/runs/detect/train2/weights/best.pt...
Ultralytics YOLOv8.0.104 🚀 Python-3.10.11 torch-2.0.0+cu118 CUDA:0 (Tesla T4, 15102MiB)
Model summary (fused): 168 layers, 11125971 parameters, 0 gradients
    Class      Images   Instances     Box(P)        R      mAP50    mAP50-95: 100% 52/52 [01:10<00:00,  1.35s/it]
        all       1642      13171      0.856       0.743      0.847      0.596
Speed: 0.5ms preprocess, 2.9ms inference, 0.0ms loss, 2.8ms postprocess per image
Results saved to /content/ultralytics/runs/detect/train2
```

Hình 18: Nội dung in màn hình sau khi kết thúc quá trình training

Kiểm tra thư mục **./runs**, các bạn sẽ thấy một file **./detect/train** xuất hiện, đây chính là file output của YOLOv8.



Hình 19: Thư mục train

7. Thực hiện Detection (Prediction) với mô hình đã huấn luyện: Để sử dụng mô hình đã huấn luyện lên một tấm ảnh bất kỳ, các bạn hãy sử dụng câu lệnh bên dưới:

```
1 # With uploaded image
2 !yolo predict model=<weight_path> source=<image_path>
```

Trong đó:

- <weight_path>: Đường dẫn đến file weight của mô hình sau khi kết thúc quá trình huấn luyện. Các bạn có thể tìm thấy trong đoạn in kết thúc quá trình huấn luyện của YOLOv8.

```
Epoch    GPU_mem    box_loss    cls_loss    dfl_loss    Instances    Size
19/20    4.05G    0.7462    0.4929    0.9263    158        640: 100% 139/139 [04:49<00:00, 2.08s/it]
          Class   Images   Instances   Box(P      R       mAP50  mAP50-95): 100% 52/52 [01:06<00:00, 1.28s/it]
          all     1642     13171     0.86      0.745    0.843      0.588

Epoch    GPU_mem    box_loss    cls_loss    dfl_loss    Instances    Size
20/20    4.07G    0.739     0.4844    0.9218    159        640: 100% 139/139 [04:48<00:00, 2.08s/it]
          Class   Images   Instances   Box(P      R       mAP50  mAP50-95): 100% 52/52 [01:16<00:00, 1.47s/it]
          all     1642     13171     0.855     0.744    0.847      0.596

20 epochs completed in 2.010 hours.
Optimizer stripped from /content/ultralytics/runs/detect/train2/weights/last.pt, 22.5MB
Optimizer stripped from /content/ultralytics/runs/detect/train2/weights/best.pt, 22.5MB

Validating /content/ultralytics/runs/detect/train2/weights/best.pt...
Ultralytics YOLOv8.0.104 🚀 Python-3.10.11 torch-2.0.0+cu118 CUDA:0 (Tesla T4, 15102MiB)
Model summary (fused): 168 layers, 11125971 parameters, 0 gradients
          Class   Images   Instances   Box(P      R       mAP50  mAP50-95): 100% 52/52 [01:10<00:00, 1.35s/it]
          all     1642     13171     0.856     0.743    0.847      0.596
Speed: 0.5ms preprocess, 2.9ms inference, 0.0ms loss, 2.8ms postprocess per image
Results saved to /content/ultralytics/runs/detect/train2
```

Đường dẫn đến file weight sau khi huấn luyện

Hình 20: Đường dẫn đến file weight của mô hình đã huấn luyện. **Lưu ý:** Các bạn cần chọn file **best.pt**

- <image_path>: Đường dẫn đến file hình ảnh input.

Dưới đây là ảnh kết quả minh họa sau khi thực thi đoạn code trên:

```
1 # With uploaded image
2 !yolo predict model=./runs/detect/train/weights/best.pt \
3 |   source='/content/ultralytics/frame007.25.00-07.30.00.jpg'
```

```
Ultralytics YOLOv8.0.104 🚀 Python-3.10.11 torch-2.0.0+cu118 CUDA:0 (Tesla T4, 15102MiB)
Model summary (fused): 168 layers, 11125971 parameters, 0 gradients

image 1/1 /content/ultralytics/frame007.25.00-07.30.00.jpg: 384x640 8 Humans, 61.4ms
Speed: 2.3ms preprocess, 61.4ms inference, 121.0ms postprocess per image at shape (1, 3, 640, 640)
Results saved to /content/ultralytics/runs/detect/predict6
```

Đường dẫn đến kết quả Detection

Bên cạnh source về ảnh, các bạn cũng có thể input source với các tham số khác nhau, đại diện cho các dữ liệu đầu vào khác nhau:

source	model(arg)	type	notes
image	'im.jpg'	str, Path	
URL	'https://ultralytics.com/images/bus.jpg'	str	
screenshot	'screen'	str	
PIL	Image.open('im.jpg')	PIL.Image	HWC, RGB
OpenCV	cv2.imread('im.jpg')	np.ndarray	HWC, BGR
numpy	np.zeros((640,1280,3))	np.ndarray	HWC
torch	torch.zeros(16,3,320,640)	torch.Tensor	BCHW, RGB
CSV	'sources.csv'	str, Path	RTSP, RTMP, HTTP
video	'vid.mp4'	str, Path	
directory	'path/'	str, Path	
glob	'path/*.jpg'	str	Use * operator
YouTube	'https://youtu.be/Zgi9g1ksQHc'	str	
stream	'rtsp://example.com/media.mp4'	str	RTSP, RTMP, HTTP

Hình 21: Tổng hợp các kiểu dữ liệu đầu vào YOLOv8 hỗ trợ trong việc thực hiện predict

Bước này cũng là bước cuối cùng trong chuỗi hướng dẫn sử dụng YOLOv8. Dối với mục tiêu của project, các bạn chỉ cần áp dụng các bước hướng dẫn trên cho dữ liệu **human** (một số bước thực hiện cần thay đổi cho phù hợp) và thực hiện đến bước số 7 là thành công.

8. **OPTIONAL:** Phần này sẽ bàn luận thêm một vài vấn đề trong YOLOv8 bao gồm một vài các tham số trong lệnh training, lệnh đánh giá mô hình và gán nhãn dữ liệu.

- **Các tham số trong lệnh training:** Dòng lệnh training tại bước 6 có mặc định săn một vài tham số, các tham số này các bạn có thể tùy chỉnh theo ý muốn của mình, với một số tham số có giá trị khác nhau sẽ cho ra hiệu suất mô hình khác nhau. Sau đây là ý nghĩa cơ bản của một vài tham số trên:
 - **img:** Kích thước ảnh training, các ảnh train và test sẽ được resize lại về kích thước bạn gán, mặc định là 640. Các bạn hoàn toàn có thể thử nghiệm trên các kích thước ảnh khác nhau.
 - **batch:** Khi thực hiện tính toán trong quá trình training, các mô hình có thể đọc một lúc toàn bộ dữ liệu train hoặc chia ra đọc theo từng batch. Với mặc định là 64, bộ dữ liệu train sẽ được chia ra thành các batch có 64 mẫu dữ liệu. Các bạn có thể cài đặt các giá trị khác theo $2^n (n \geq 0)$.
 - **epochs:** Số lần lặp qua bộ dữ liệu trong quá trình huấn luyện.
 - **data:** Thông tin bộ dữ liệu (file .yaml) mà bạn mong muốn training.
 - **weights:** File pretrained model sử dụng. Các bạn có thể tải và sử dụng các file pretrained model khác nhau trong danh sách [này](#).
- **Đánh giá mô hình:** Như đã nói ở phần trước, hiệu suất của mô hình có thể thay đổi với các giá trị tham số khác nhau. Để có tìm ra mô hình tốt nhất một cách định lượng, ta có thể thực hiện dòng lệnh sau:

```
1 !yolo val model=./runs/detect/train2/weights/best.pt data=../human_detection_dataset/data.yaml

Ultralytics YOLOv8.0.104 🚀 Python-3.10.11 torch-2.0.0+cu118 CUDA:0 (Tesla T4, 15102MiB)
Model summary (fused): 168 layers, 11125971 parameters, 0 gradients
val: Scanning /content/human_detection_dataset/val/labels.cache... 1642 images, 0 backgrounds, 0 corrupt: 100%
    Class      Images   Instances     Box(P       R       mAP50   mAP50-95): 100% 103/103 [01:19<
        all       1642     13171      0.857     0.743     0.847     0.598
Speed: 0.4ms preprocess, 5.6ms inference, 0.0ms loss, 2.7ms postprocess per image
Results saved to /content/ultralytics/runs/detect/val
```

Hình 22: Kết quả đánh giá trên bộ val

- **Gán nhãn dữ liệu:** Vì YOLOv8 là học có giám sát, tức các mẫu trong bộ dữ liệu training cần phải có labels tương ứng với từng mẫu. Vì vậy, để có thể cung cấp thêm dữ liệu hoặc tạo ra một bộ dữ liệu mới, với các class mới. Ta cần phải thực hiện gán nhãn dữ liệu, việc gán nhãn này sẽ phải thực hiện thủ công. Trong bài hướng dẫn này, chúng ta sẽ tìm hiểu cách sử dụng [labelImg](#). Để cài đặt labelImg về máy tính, có rất nhiều cách khác nhau (các bạn có thể đọc file README.md trên trang GitHub của labelImg), ở đây chúng ta sẽ chọn cách cài đặt với Anaconda:

- **Bước 1:** Tải mã nguồn của labelImg tại trang GitHub sử dụng dòng lệnh sau (các dòng lệnh trong các bước ở đây sẽ được thực hiện trong cmd/terminal):

```
1 $ git clone https://github.com/heartexlabs/labelImg.git
```

- **Bước 2:** Cài đặt Anaconda tại [đây](#).

- **Bước 3:** Sau khi đã cài đặt Anaconda, các bạn sẽ khởi tạo môi trường Anaconda bằng dòng lệnh sau:

```
1 $ conda create -n labelimg_env python=3.9 -y
```

- **Bước 4:** Kích hoạt môi trường ảo đã tạo ở bước 3 sử dụng lệnh:

```
1 $ conda activate labelimg_env
```

- **Bước 5:** Di chuyển vào thư mục mã nguồn của labelImg sử dụng lệnh sau:

```
1 $ cd <path_to_labelImg_folder>
```

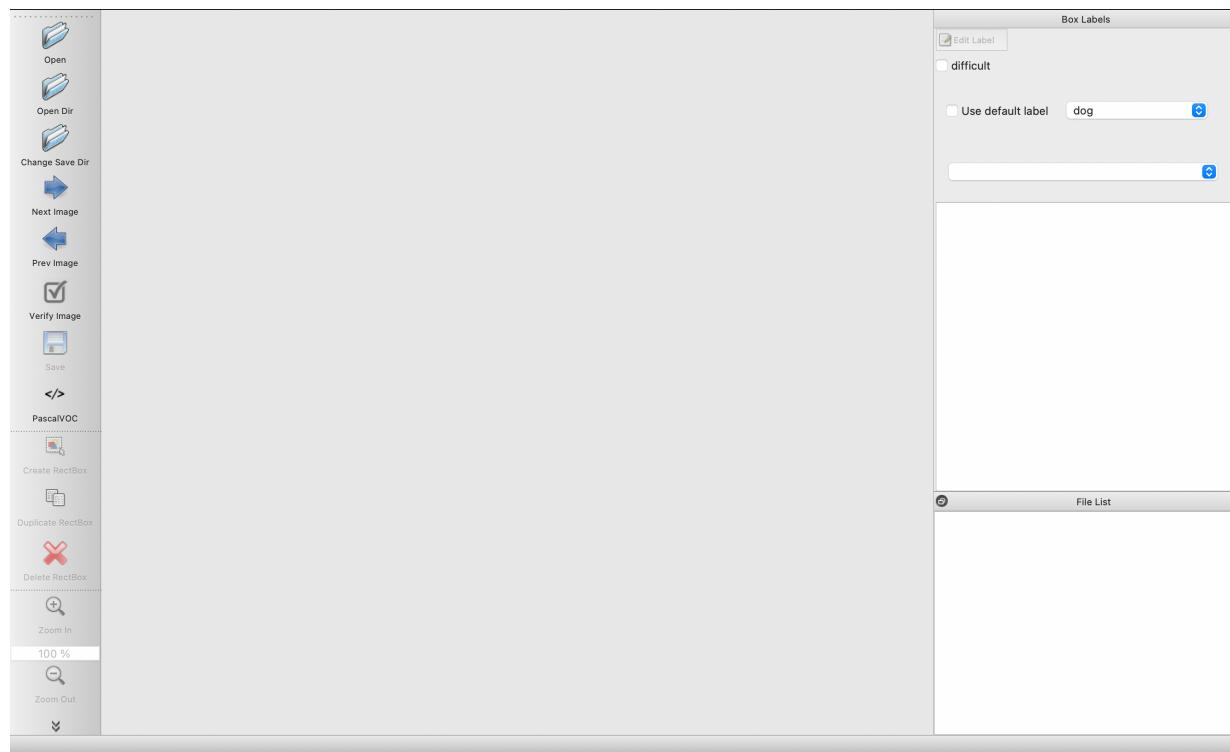
- **Bước 6:** Thực hiện lần lượt các dòng lệnh:

```
1 $ conda install pyqt=5
2 $ conda install -c anaconda lxml
3 $ pyrcc5 -o libs/resources.py resources.qrc
```

- **Bước 7:** Khởi động giao diện của labelImg sử dụng lệnh:

```
1 $ python labelImg.py
```

Nếu thành công, các bạn sẽ thấy một giao diện màn hình như sau:



Về sau, mỗi lần cần sử dụng labelImg, ta chỉ cần kích hoạt môi trường đã tạo này và khởi động labelImg (tức chỉ thực hiện bước 4 và bước 7).

Bây giờ, giả sử ta muốn gán nhãn một tập ảnh người (class person). Chúng ta sẽ thực hiện các bước như sau (chuẩn bị sẵn một thư mục hình ảnh):

- **Bước 1:** Truy cập vào thư mục data trong mã nguồn labelImg, sửa lại nội dung file **predefined_classes.txt** bằng tên của các class trong bộ dữ liệu của bạn, mỗi tên class sẽ là một hàng trong file:

```

predefined_classes.txt
dog
person
cat
tv
car
meatballs
marinara sauce
tomato soup
chicken noodle soup
french onion soup
chicken breast
ribs
pulled pork
hamburger
cavity

```

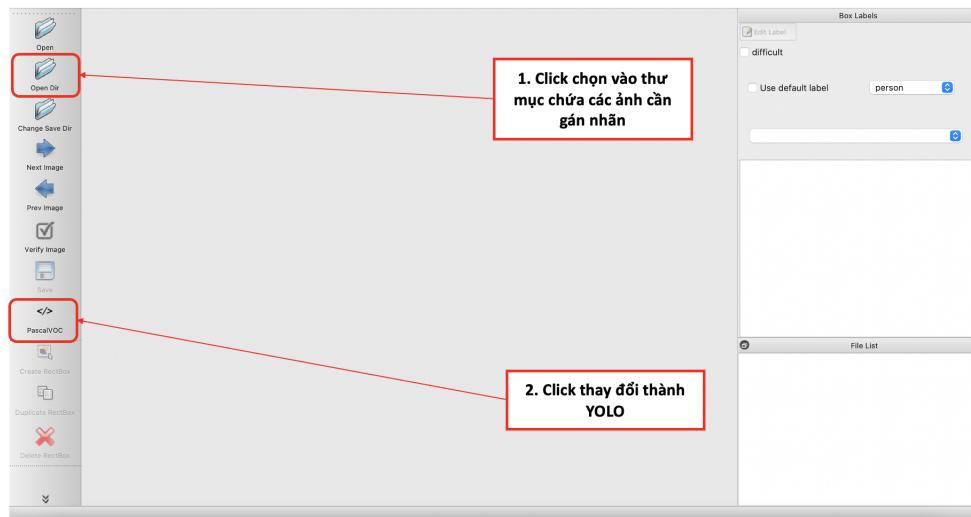
Hình 23: Nội dung trong file **predefined_classes.txt**. Mỗi hàng trong file tương ứng với mỗi tên class trong bộ dữ liệu.

Các bạn sẽ sửa lại nội dung file bằng tên của class trong bộ dữ liệu của mình, giả sử trong bộ **human** chỉ có class là person, vậy ta chỉ sửa lại file thành như sau:



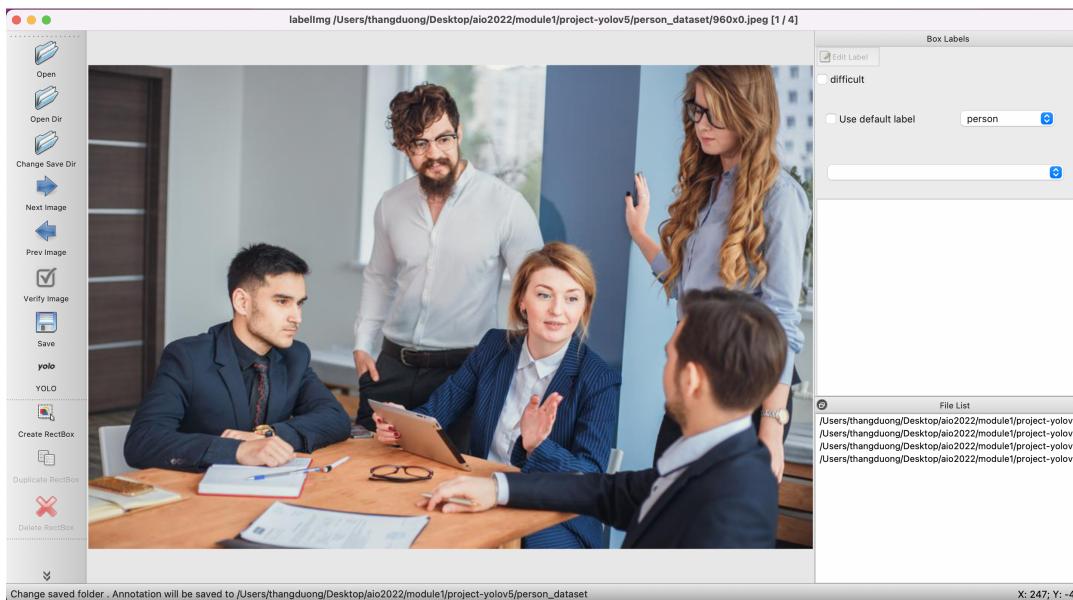
Hình 24: Nội dung trong file **predefined_classes.txt** với bộ dữ liệu **human**.

– **Bước 2:** Truy cập vào giao diện labelImg, các bạn hãy làm theo các bước theo hình sau:



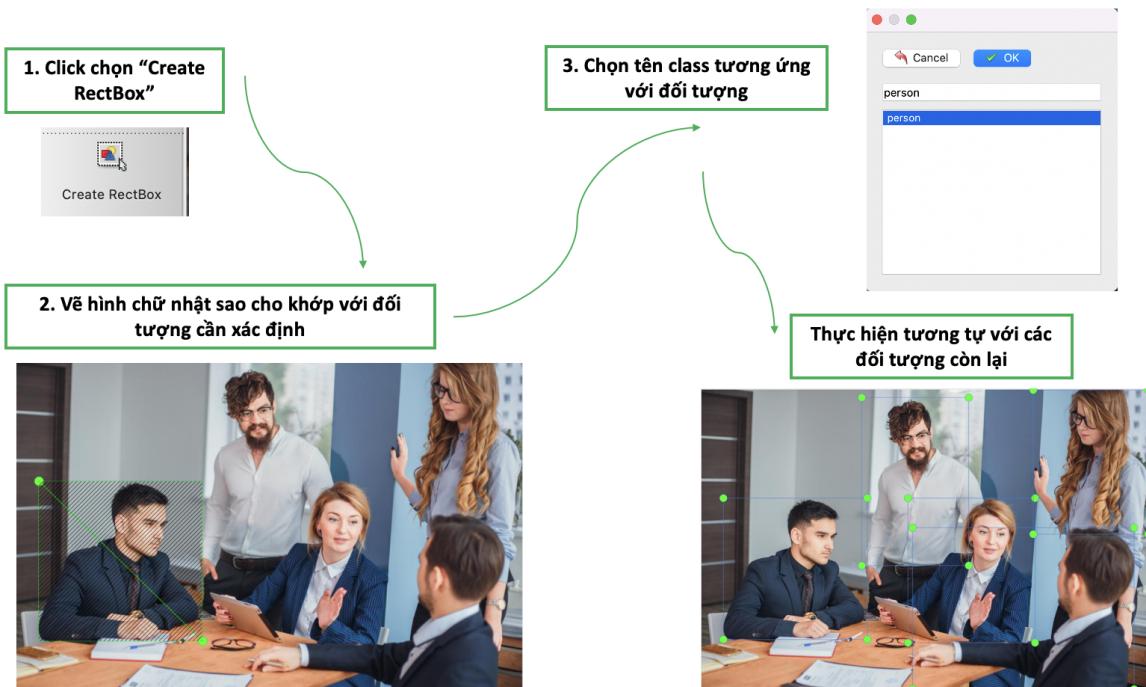
Hình 25: Mở thư mục chứa ảnh cần gán nhãn và chuyển format nhãn thành YOLO

– **Bước 3:** Nếu thành công, các bạn sẽ thấy giao diện như sau:



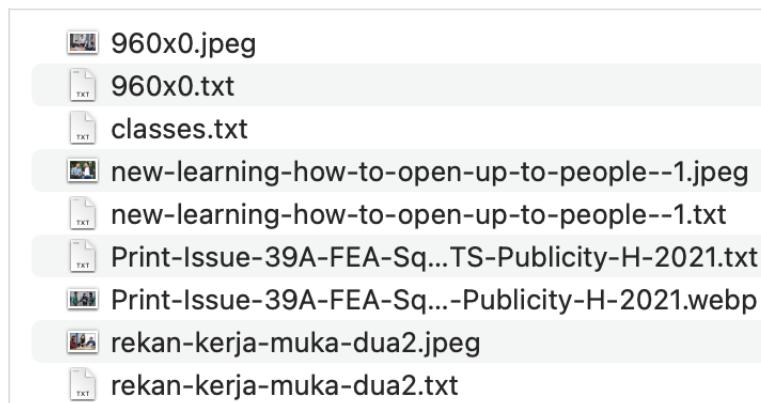
Hình 26: Giao diện sau khi chọn thư mục ảnh cần gán nhãn

– **Bước 4:** Thực hiện gán nhãn. Các bạn làm theo hình dưới đây:



Hình 27: Quá trình gán nhãn

Sau khi kết thúc gán nhãn cho một ảnh, các bạn save lại (CTRL + S) và di chuyển đi hình tiếp theo (phím D (Next Image) hoặc phím A (Prev Image)). Khi đã gán nhãn hết tất cả các ảnh, các bạn có thể kiểm tra tại thư mục chứa ảnh của mình, nếu thấy các file .txt cho từng ảnh bạn đã gán nhãn, bạn đã gán nhãn thành công:

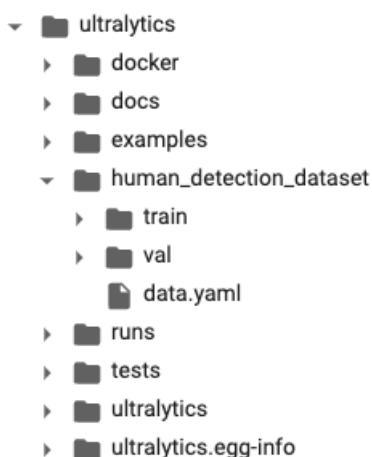


Hình 28: Thư mục chứa ảnh sau khi gán nhãn

Đây là công việc tưởng chừng nhẹ nhàng nhưng lại hết sức quan trọng, vì nó sẽ ảnh hưởng rất lớn đến kết quả từ mô hình của bạn, vì vậy cần thật sự tập trung và cẩn trọng trong việc gán nhãn dữ liệu.

Phần III: Trắc nghiệm

1. Trong một chương trình Human Detection với chức năng dùng để nhận diện người có trong ảnh, Input của chương trình là gì?
 - (a) Mô hình Object Detection.
 - (c) Ảnh chứa người.
 - (b) Ảnh bất kì.
 - (d) Bounding Box.
2. Trong một chương trình Object Detection, Output của mô hình Object Detection là gì?
 - (a) Tọa độ bounding box và tên class nếu có.
 - (c) Tọa độ bounding box nếu có.
 - (b) Ảnh chứa người.
 - (d) Ảnh chứa bounding box nếu có.
3. Trong các bước cài đặt và thực hiện huấn luyện YOLOv8, bước nào sau đây không nằm trong quy trình?
 - (a) Tải pretrained model.
 - (c) Tải bộ dữ liệu.
 - (b) Cài đặt thư viện ultralytics.
 - (d) Cài đặt hàm huấn luyện.
4. Trong các bước sử dụng labelImg để gán nhãn dữ liệu huấn luyện YOLOv8, bước nào sau đây không nằm trong thao tác chính?
 - (a) Chọn thư mục chứa ảnh.
 - (c) Đổi format label sang PascalVOC.
 - (b) Chọn thư mục lưu label.
 - (d) Sửa tên class trong `predefine_classes.txt`
5. Trong file cấu hình của bộ dữ liệu (file `.yaml`), trường thông tin **nc** có ý nghĩa gì?
 - (a) Số mẫu dữ liệu trong dataset.
 - (c) Số lượng dataset.
 - (b) Số lượng class.
 - (d) Số thứ tự của dataset.
6. Lệnh nào sau đây dùng để kích hoạt môi trường ảo tên `yolo_env` trong conda?
 - (a) `conda deactivate yolo_env`
 - (c) `conda env activate yolo_env`
 - (b) `conda env deactivate yolo_env`
 - (d) `conda activate yolo_env`
7. Cho hình ảnh cấu trúc cây thư mục (trong Google Colab) và đoạn lệnh sau:



```
1 !yolo train model=yolov8s.pt data=../human_detection_dataset/data.yaml epochs=20  
      imgsz=640
```

Giả sử các bạn đang ở thư mục **/content/ulytralytics**, khi thực thi đoạn lệnh trên có xảy ra lỗi không?

- (a) Có. (b) Không.
8. Các bạn hãy sử dụng mô hình đã huấn luyện được, thực hiện predict trên video youtube [này](#) và submit link video (link youtube, drive...) kết quả các bạn đạt được.

- *Hết* -