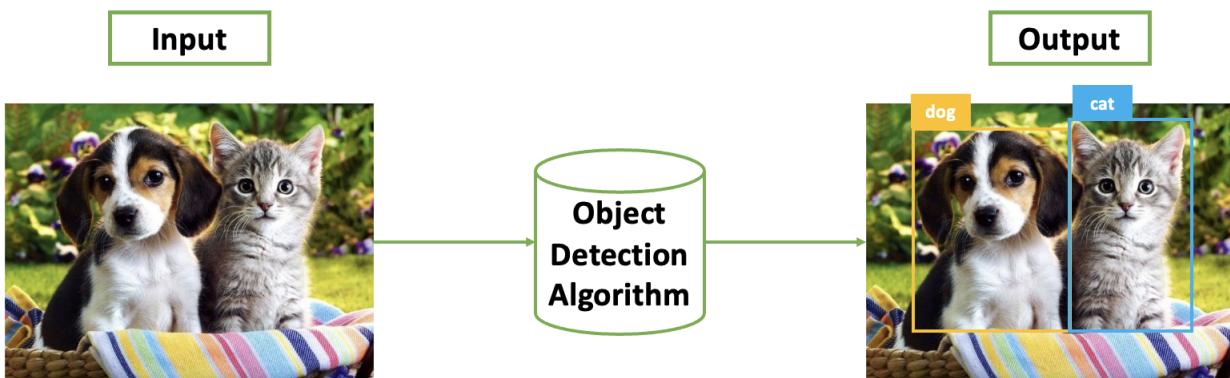


AI VIET NAM – COURSE 2022

Module 1 – Project

Ngày 9 tháng 7 năm 2022

Phát hiện đối tượng (Object Detection) là một bài toán trong lĩnh vực Thị giác máy tính (Computer Vision), trong đó nhiệm vụ của chúng ta là viết 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 xác định.



Hình 1: Input/Output của bài toán Object Detection

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 **YOLOv5**.



(a) Ảnh Input

(b) Kết quả Detection của YOLOv5

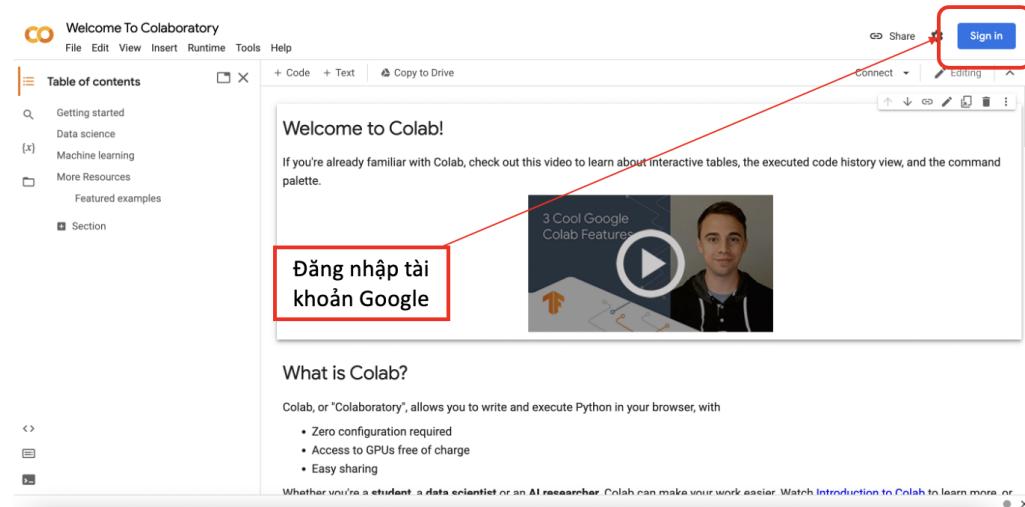
Hình 2: Ví dụ minh họa kết quả của mô hình Human Detection

Yêu cầu project: các bạn sẽ đọc, tìm hiểu cách sử dụng mã nguồn YOLOv5 (có hướng dẫn bên dưới) và áp dụng vào bộ dữ liệu **human** (tải tại [đây](#)).

Hướng dẫn sử dụng YOLOv5: Để cài đặt và sử dụng thuật toán YOLOv5, các bạn hãy thực hiện theo các bước hướng dẫn sau (bài hướng dẫn sẽ sử dụng dữ liệu mẫu khác với dữ liệu yêu cầu project để làm ví dụ).

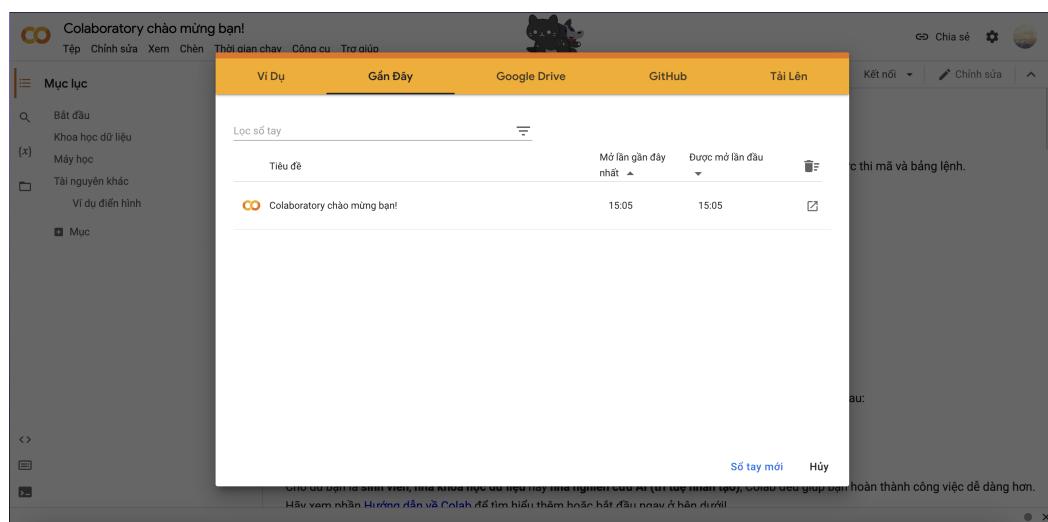
1. **Chuẩn bị môi trường cài đặt:** Để thuận tiện trong việc sử dụng YOLOv5, chúng ta sẽ dùng Google Colab làm môi trường cài đặt. Các bước sử dụng Google Colab được thực hiện như sau:

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



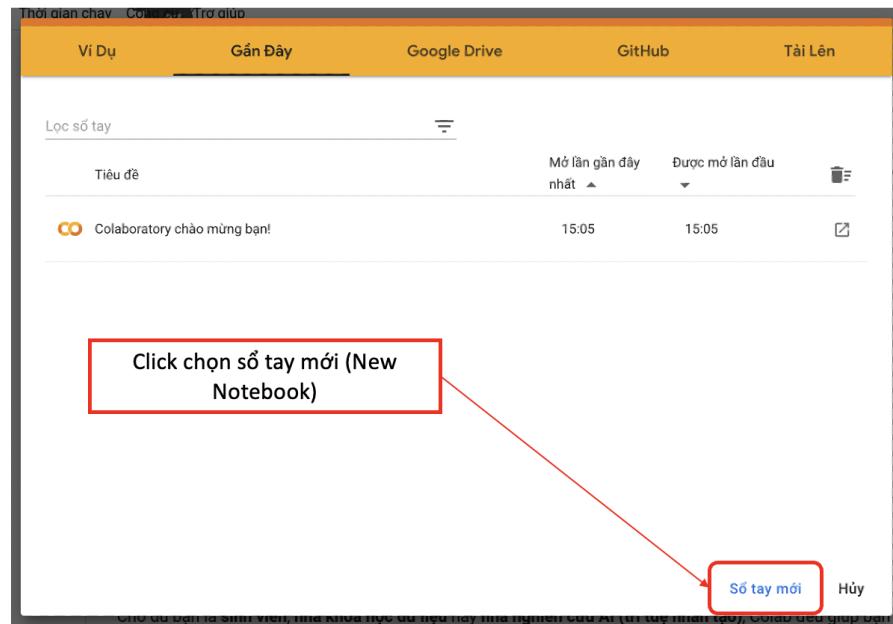
Hình 3: Giao diện chính của Google Colab

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:



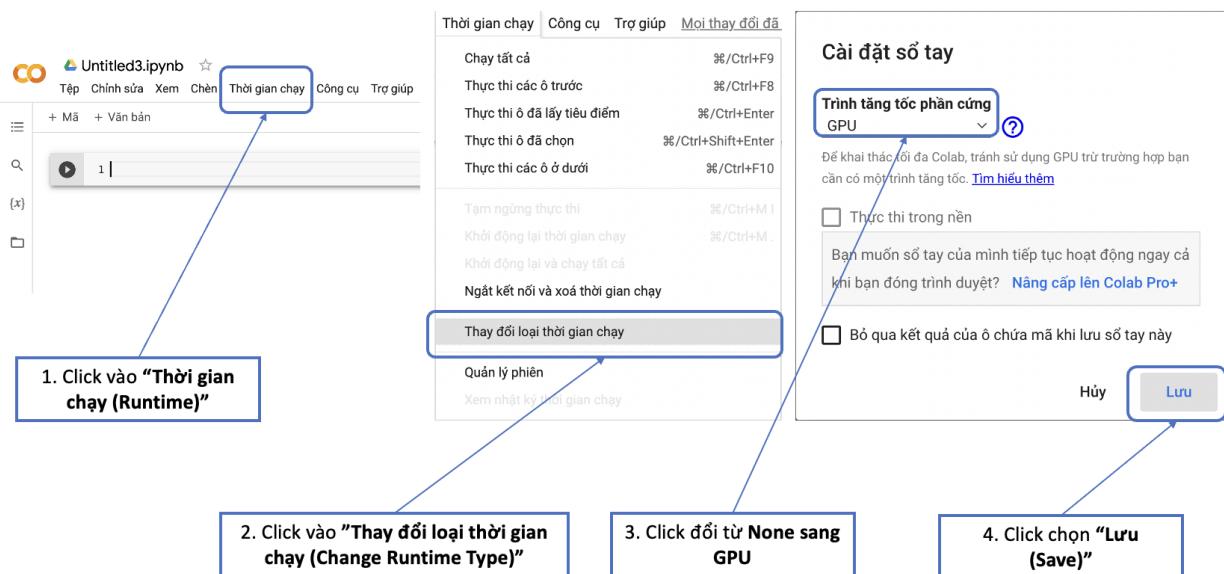
Hình 4: Giao diện Google Colab sau khi đăng nhập thành công

- **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:



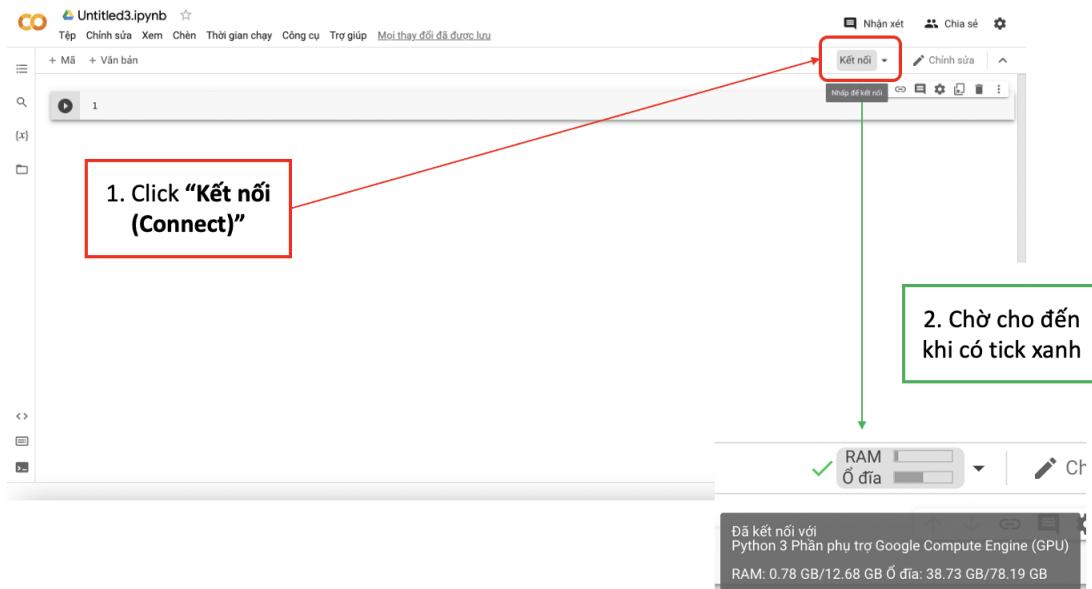
Hình 5: Khởi tạo notebook mới

- 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:

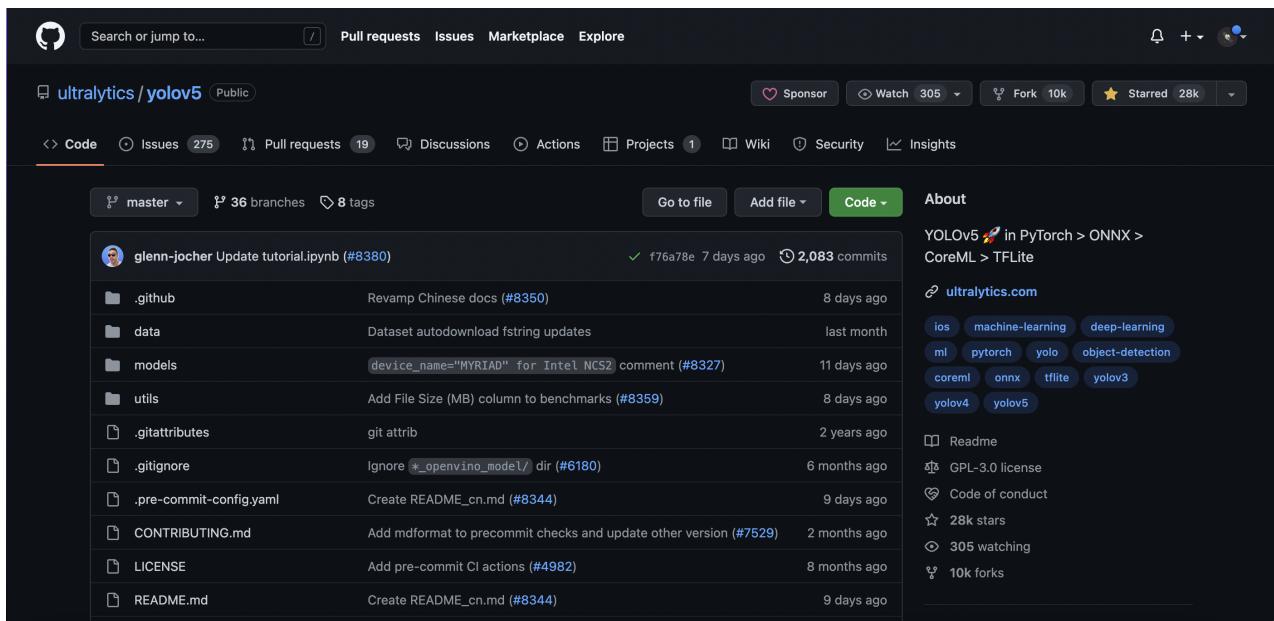


Hình 7: Khởi động một notebook có GPU trong Google Colab

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

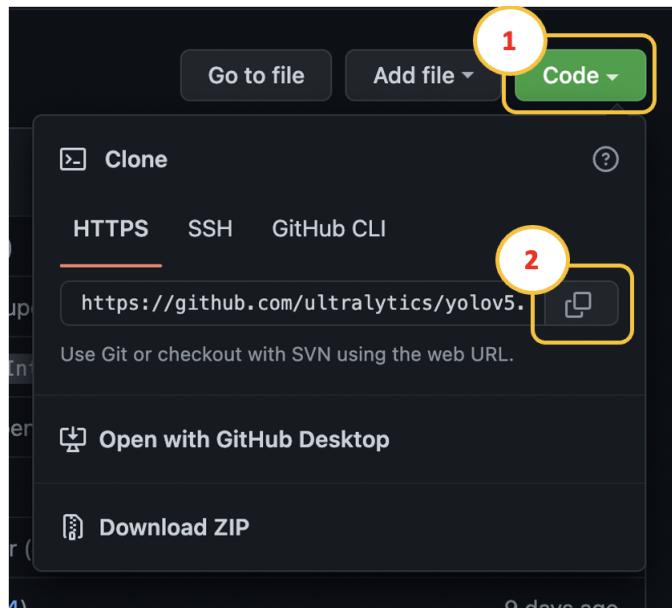
2. **Tải mã nguồn YOLOv5 từ GitHub:** Để sử dụng YOLOv5, chúng ta cần tải mã nguồn (source code) của YOLOv5 về môi trường cài đặt code, mã nguồn của YOLOv5 được công khai trên GitHub. Như vậy, các bạn sẽ thực hiện theo các bước sau:

- **Bước 1:** Các bạn truy cập vào đường dẫn GitHub của YOLOv5 tại [đây](#). Nếu truy cập thành công, các bạn sẽ thấy giao diện như hình dưới:



Hình 8: Giao diện GitHub của YOLOv5

- **Bước 2:** Tại trang GitHub của YOLOv5, các bạn chọn mục **Code** (có màu nền xanh lá) và chọn nút copy đường dẫn như ảnh minh họa dưới đây:



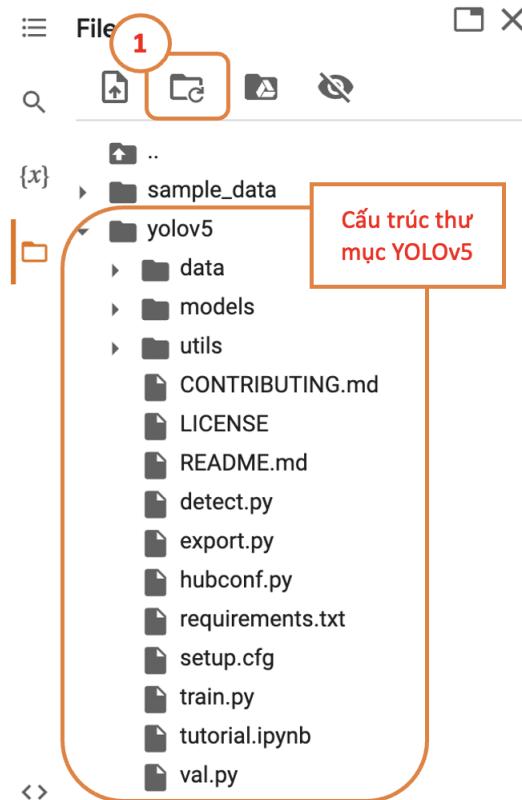
Hình 9: Copy đường dẫn để tải mã nguồn YOLOv5

- **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]  1 !git clone https://github.com/ultralytics/yolov5.git
Cloning into 'yolov5'...
remote: Enumerating objects: 12388, done.
remote: Counting objects: 100% (8/8), done.
remote: Compressing objects: 100% (8/8), done.
remote: Total 12388 (delta 1), reused 6 (delta 0), pack-reused 12380
Receiving objects: 100% (12388/12388), 12.16 MiB | 5.15 MiB/s, done.
Resolving deltas: 100% (8544/8544), done.
```

Hình 10: Tải mã nguồn YOLOv5 sử dụng lệnh git clone

Để kiểm tra, các bạn có thể refresh lại phần **Files** của Google Colab để xem thư mục yolov5 đã xuất hiện hay chưa.



Hình 11: Thư mục yolov5

- Cài đặt các gói thư viện Python cần thiết theo yêu cầu của YOLOv5: Mã nguồn YOLOv5 được xây dựng bằng rất nhiều các thư viện Python khác nhau. Vì vậy, để có thể chạy được YOLOv5 ta cần tải các gói thư viện cần thiết mà YOLOv5 yêu cầu bằng cách chạy hai câu lệnh trong ảnh sau:

```

1 %cd yolov5
2 !pip install -r requirements.txt -q

```

/content/yolov5 | 596 kB 4.7 MB/s

Hình 12: Tải các gói thư viện cần thiết dùng lệnh pip

- Tải file pretrained model: Các bạn tải file pretrained model tại [đây](#) và đặt file vào thư mục yolov5. Trên Google Colab, việc này có thể được thực hiện thông qua lệnh wget như hình dưới đây:

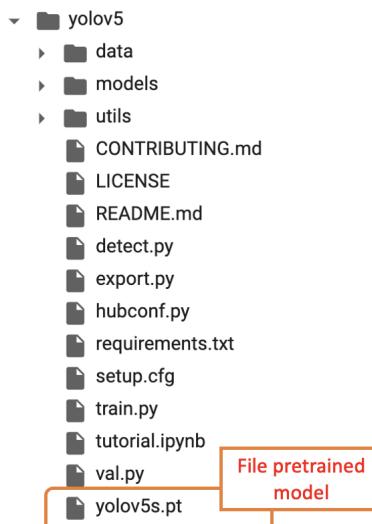
```
1 !wget https://github.com/ultralytics/yolov5/releases/download/v6.1/yolov5s.pt
--2022-07-05 10:05:41-- https://github.com/ultralytics/yolov5/releases/download/v6.1/yolov5s.pt
Resolving github.com (github.com)... 20.205.243.166
Connecting to github.com (github.com)|20.205.243.166|:443... connected.
HTTP request sent, awaiting response... 302 Found
Location: https://objects.githubusercontent.com/github-production-release-asset-2e65be/264818686/
--2022-07-05 10:05:41-- https://objects.githubusercontent.com/github-production-release-asset-2e65be/264818686/
Resolving objects.githubusercontent.com (objects.githubusercontent.com)... 185.199.108.133, 185.1
Connecting to objects.githubusercontent.com (objects.githubusercontent.com)|185.199.108.133|:443...
HTTP request sent, awaiting response... 200 OK
Length: 14808437 (14M) [application/octet-stream]
Saving to: 'yolov5s.pt'

yolov5s.pt      100%[=====] 14.12M  --.-KB/s    in 0.05s

2022-07-05 10:05:41 (309 MB/s) - 'yolov5s.pt' saved [14808437/14808437]
```

Hình 13: Tải file pretrained model sử dụng lệnh wget

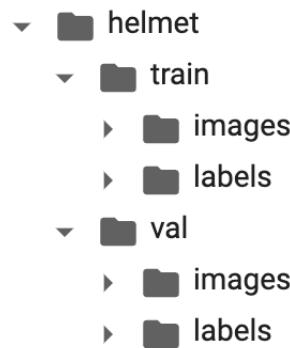
Để kiểm tra xem file pretrained model đã được tải về thành công hay chưa, các bạn refresh phần **Files** của Google Colab và tìm kiếm file có tên **yolov5s.pt**:



Hình 14: File pretrained model

5. **Chuẩn bị dữ liệu:** để huấn luyện YOLOv5 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 **yolov5**) có cấu trúc cây thư mục như sau:



Hình 15: Tổ chức thư mục bộ dữ liệu trong YOLOv5

Trong đó:

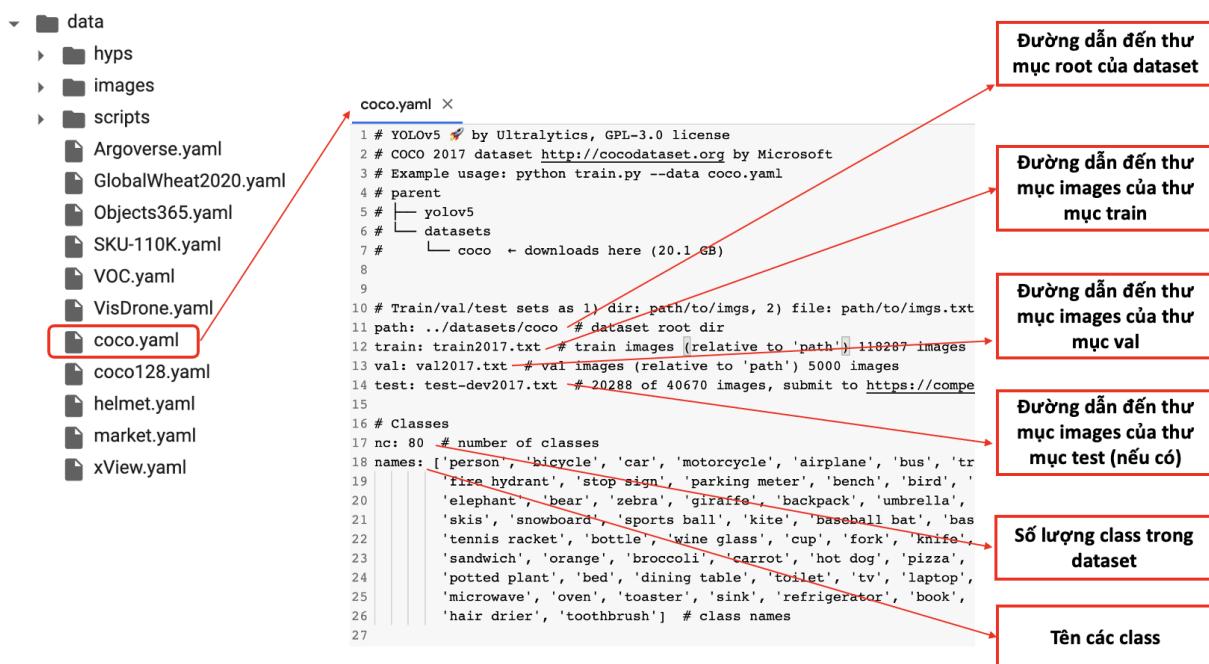
- **Thư mục images:** chứa các file ảnh (.jpg, .png...).
- **Thư mục labels:** chứa các file .txt tương ứng với từng ảnh (có cùng tên file) trong thư mục images.

Phần hướng dẫn này sẽ sử dụng bộ dữ liệu **Helmet** làm ví dụ (các bạn có thể tải bộ dữ liệu này 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 # move to yolov5 folder first and then:
2 !gdown --id 1bYytJdFa1D_r2km6ZZ8T99TnPlvyCAhf
3 !unzip helmet.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 YOLOv5 cung cấp sẵn:



Hình 16: Các trường thông tin quan trọng trong file .yaml

Như vậy, đối với bộ dữ liệu **Helmet**, ta chỉ việc tạo một file .yaml mới (ví dụ helmet.yaml) và điền các trường thông tin tương ứng. 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 phải được đặt ở trong thư mục **data**):

```

1 import yaml
2
3 dataset_info = {
4     'path': 'helmet',
5     'train': 'train/images',
6     'val': 'val/images',
7     'nc': 3,
8     'names': ['helmet', 'head', 'person']
9 }
10
11 with open('data/helmet.yaml', 'w+') as f:
12     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...).

- 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 !python train.py --img 640 --batch 64 --epochs 20 --data helmet.yaml --weights
    yolov5s.pt --cache

```

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

```

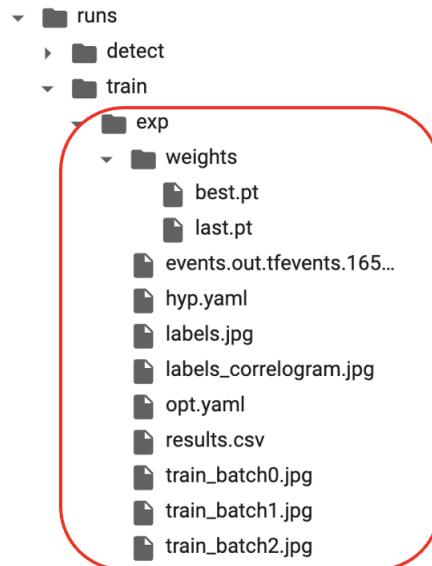
20 epochs completed in 0.515 hours.
Optimizer stripped from runs/train/exp/weights/last.pt, 14.5MB
Optimizer stripped from runs/train/exp/weights/best.pt, 14.5MB

Validating runs/train/exp/weights/best.pt...
Fusing layers...
Model summary: 213 layers, 7018216 parameters, 0 gradients, 15.8 GFLOPs
      Class      Images      Labels      P      R      mAP@.5  mAP@.5:.95:  12% 1/8 [00:02<00:15,  2.23

```

Hình 17: Nội dung in màn hình sau khi kết thúc quá trình training

Kiểm tra thư mục **runs**, các bạn sẽ thấy một file **exp** xuất hiện, đây chính là file output của YOLOv5.



Hình 18: Thư mục exp

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

```
! python detect.py --weights "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 YOLOv5.

```
Epoch    gpu_mem      box      obj      cls      labels    img_size
18/19    13.3G    0.0296   0.05028      0       660      640: 100% 35/35 [00:41<00:00,  1.17s/it]
          Class     Images     Labels      P        R      mAP@.5 mAP@.5:.95: 100% 13/13 [00:11<00:00
          all      1642      13171      0.88      0.741      0.841      0.538

Epoch    gpu_mem      box      obj      cls      labels    img_size
19/19    13.3G    0.02952   0.05032      0       744      640: 100% 35/35 [00:41<00:00,  1.17s/it]
          Class     Images     Labels      P        R      mAP@.5 mAP@.5:.95: 100% 13/13 [00:11<00:00
          all      1642      13171      0.877      0.733      0.838      0.54

20 epochs completed in 0.295 hours.
Optimizer stripped from runs/train/exp/weights/last.pt, 14.4MB
Optimizer stripped from runs/train/exp/weights/best.pt, 14.4MB
Đường dẫn đến file weight
Validating runs/train/exp/weights/best.pt...
Fusing layers...
Model summary: 213 layers, 7012822 parameters, 0 gradients, 15.8 GFLOPs
          Class     Images     Labels      P        R      mAP@.5 mAP@.5:.95: 100% 13/13 [00:17<00:00
          all      1642      13171      0.877      0.733      0.838      0.54
Results saved to runs/train/exp
```

Hình 19: Đường dẫn đến file weight của mô hình đã huấn luyện. **Lưu ý:** các bạn nên 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:

```
[17] 1 !python detect.py --weights runs/train/exp/weights/best.pt --source mom.png
detect: weights=['runs/train/exp/weights/best.pt'], source=mom.png, data=data/coco128.yaml, imgsz=[640, 640]
YOLOv5 🚀 v6.1-289-g526e650 Python-3.7.13 torch-1.11.0+cu113 CUDA:0 (Tesla P100-PCIE-16GB, 16281MiB)

Fusing layers...
Model summary: 213 layers, 7012822 parameters, 0 gradients, 15.8 GFLOPs
image 1/1 /content/yolov5/mom.png: 448x640 3 persons, Done. (0.011s)
Speed: 0.5ms pre-process, 11.0ms inference, 1.2ms NMS per image at shape (1, 3, 640, 640)
Results saved to runs/detect/exp9
```

Đường dẫn chứa 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:

```
1 # video
2 !python detect.py --weights "runs/train/exp/weights/best.pt" --source "vid_name.
  mp4"
3 # a folder contains images
4 !python detect.py --weights "runs/train/exp/weights/best.pt" --source "
  folder_name/"
5 # youtube video
6 !python detect.py --weights "runs/train/exp/weights/best.pt" --source "https://
  www.youtube.com/watch?v=SX_ViT4Ra7k"
7 # on webcam (not working on Google Colab)
8 !python detect.py --weights "runs/train/exp/weights/best.pt" --source 0
```

Bước này cũng là bước cuối cùng trong chuỗi hướng dẫn sử dụng YOLOv5 trên dữ liệu **Helmet**. Đối 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 YOLOv5 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, mặc định sẽ lặp 20 lần.
- **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 !python val.py --weights runs/train/exp2/weights/best.pt --data helmet.yaml --img 640 --iou 0.65 --half
val: data=/content/yolov5/data/helmet.yaml, weights=['runs/train/exp2/weights/best.pt'], batch_size=32, imgsz=640,
YOLOv5 🚀 v6.1-289-g526e650 Python-3.7.13 torch-1.11.0+cu113 CUDA:0 (Tesla P100-PCIE-16GB, 16281MiB)

Fusing layers...
Model summary: 213 layers, 7018216 parameters, 0 gradients, 15.8 GFLOPs
val: Scanning '/content/yolov5/helmet/val/labels.cache' images and labels... 1000 found, 0 missing, 0 empty, 0 corr
    Class      Images     Labels      P      R   mAP@.5:m:.95: 100% 32/32 [00:17<00:00,  1
      all       1000      5104     0.943     0.594     0.626     0.408
    helmet      1000      3762     0.943     0.896     0.959     0.624
      head      1000      1201     0.884     0.885     0.91      0.596
    person      1000      141      1         0     0.00967     0.00441
Speed: 0.3ms pre-process, 4.1ms inference, 1.8ms NMS per image at shape (32, 3, 640, 640)
Results saved to runs/val/exp4

```

Hình 20: Kết quả đánh giá trên bộ val

- **Gán nhãn dữ liệu:** Vì YOLOv5 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 (ví dụ tạo thêm dữ liệu human, helmet...) 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/tzutalin/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:** Duy 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:

```

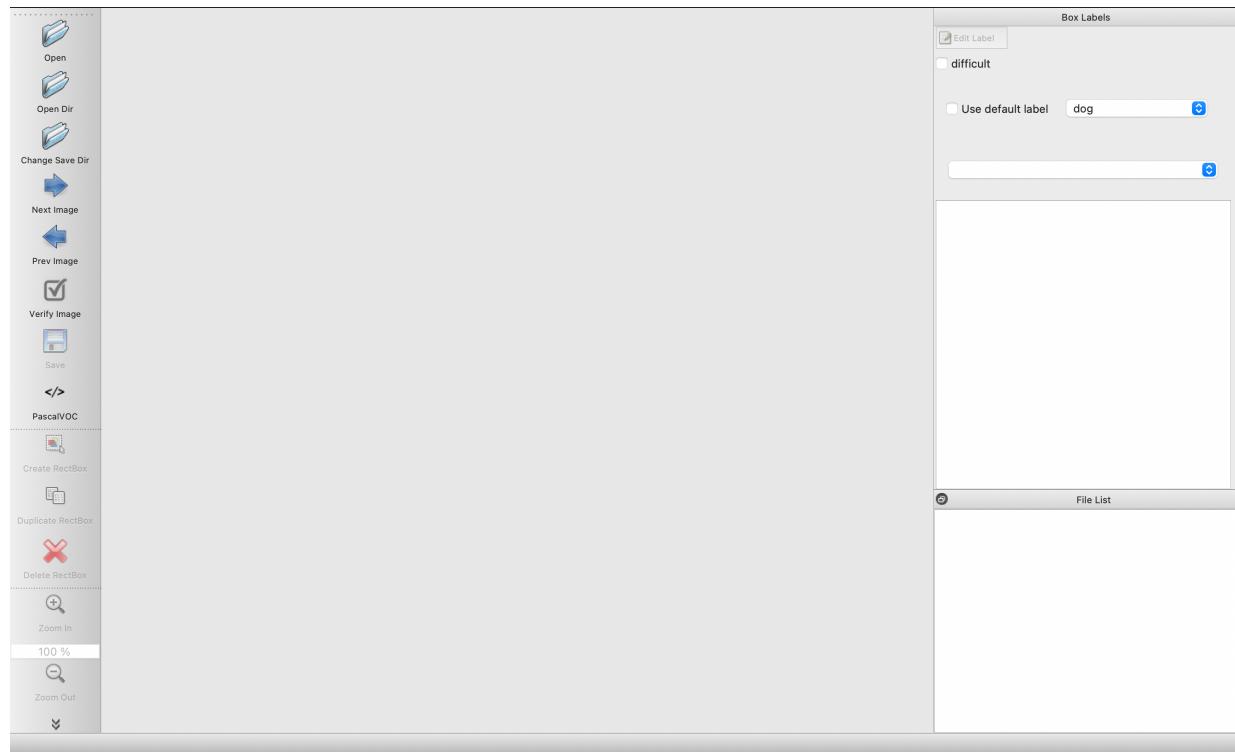
1 conda install pyqt=5
2 conda install -c anaconda lxml
3 pyrcc5 -o libs/resources.py resources.qrc

```

- **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

```

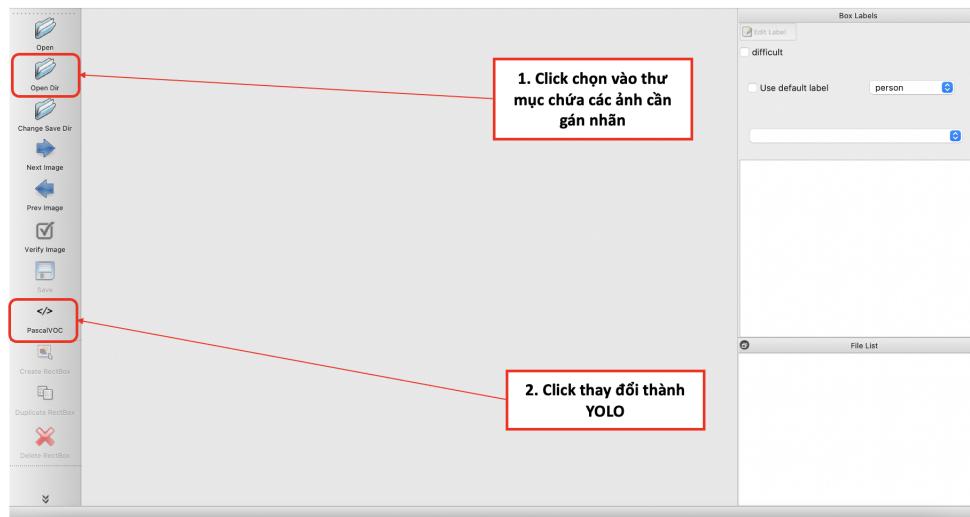
Hình 21: Nội dung trong file predefined_classes.txt. Mỗi hàng trong file tương ứng với mỗi tên class trong bộ dữ liệu.

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:



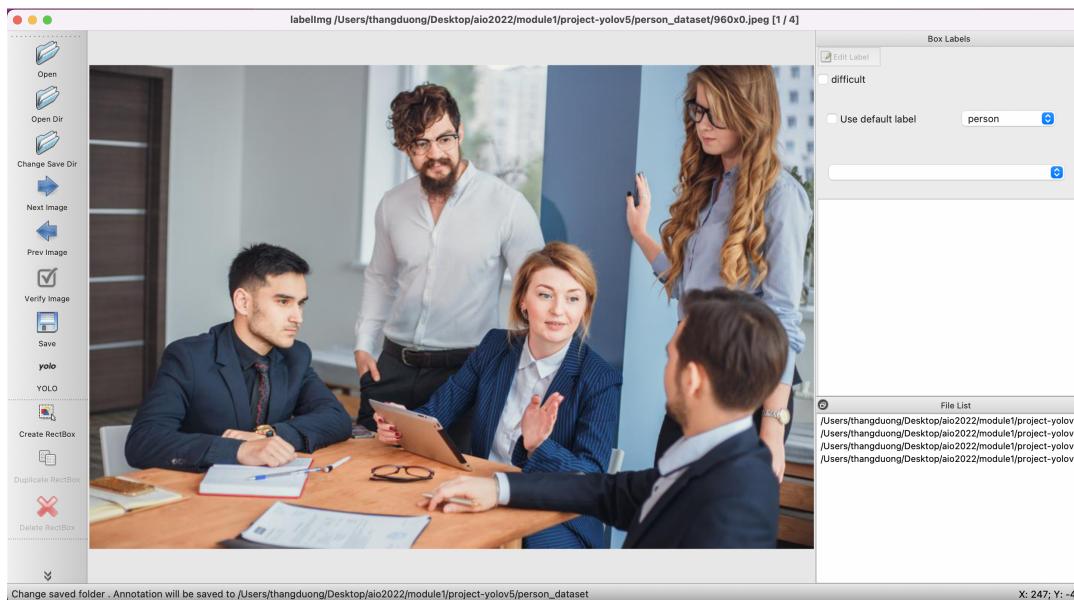
Hình 22: Nội dung trong file predefined_classes.txt với bộ dữ liệu **human**.

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



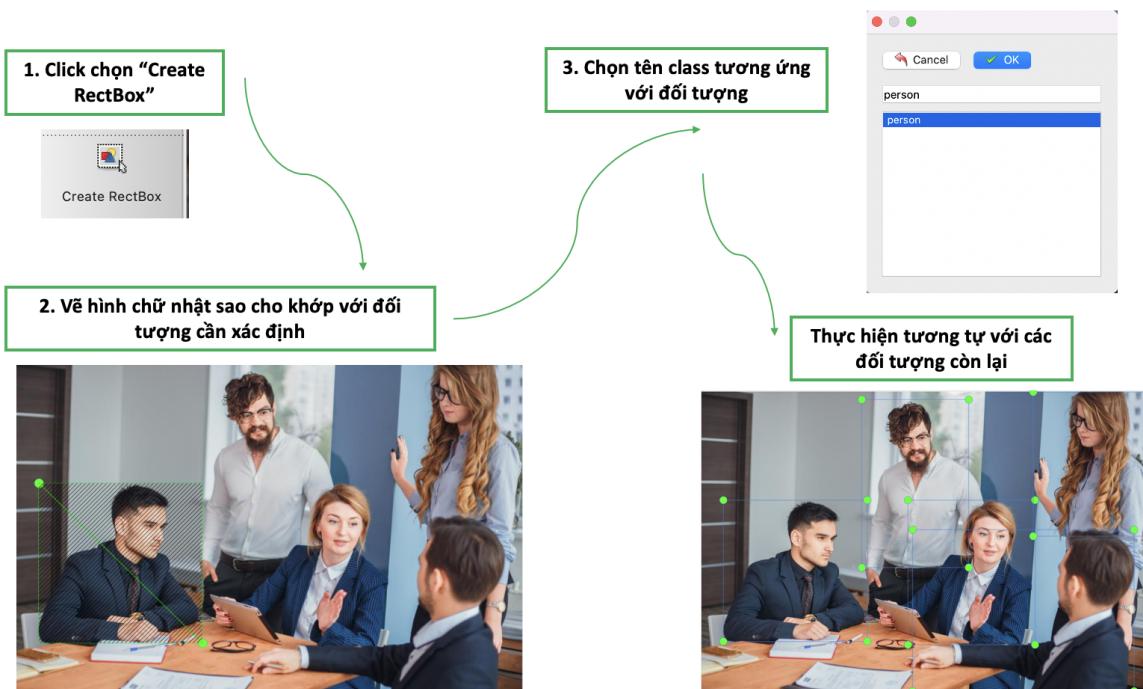
Hình 23: Mở thư mục chứa ảnh cần gán nhãn và chuyển format nhãn thành YOLO

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



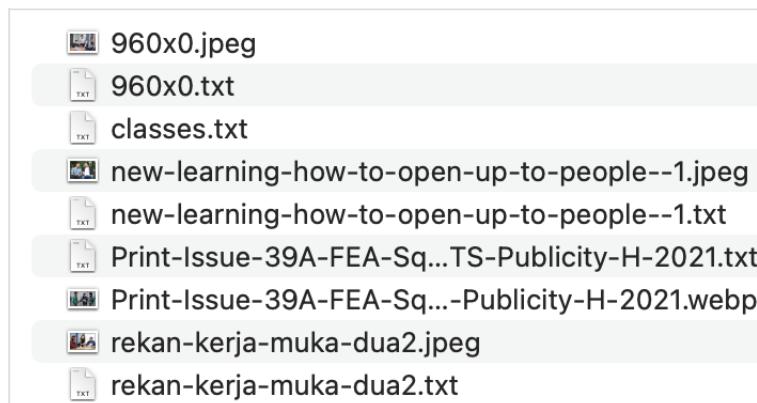
Hình 24: Giao diện sau khi chọn thư mục ảnh cần gán nhãn

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



Hình 25: 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 26: 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.

- *Hết* -