

TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN – ĐHQG TPHCM  
KHOA CÔNG NGHỆ THÔNG TIN



## BÁO CÁO ĐỒ ÁN

MÔN HỌC: HỌC THỐNG KÊ

Họ và tên: Nguyễn Doãn Tiên Đạt  
MSSV: 1712330

## Mục lục

I. THÔNG TIN VỀ MODEL .....	3
II. CHỨC NĂNG CỦA HỆ THỐNG .....	3
1. Đổi với client.....	3
2. Đổi với server .....	5
III. CÀI ĐẶT YOLOv4 TRÊN TENSORFLOW .....	6
1. Cài đặt trên PyCharm .....	6
2. Cài đặt trên Google Colab .....	9
IV. HUẤN LUYỆN YOLOv4 TRÊN TẬP DỮ LIỆU TÙY CHỈNH .....	11
1. Bài toán .....	11
2. Chuẩn bị dữ liệu .....	12
3. Tiến hành huấn luyện.....	15
4. Sử dụng trọng số huấn luyện.....	16
V. XÂY DỰNG FLASK API CHO MÔ HÌNH .....	17
VI. XÂY DỰNG GIAO DIỆN NGƯỜI DÙNG.....	20
VII. NGUỒN THAM KHẢO VÀ TÀI NGUYÊN .....	24

## I. THÔNG TIN VỀ MODEL

- Mô hình sử dụng: YOLOv4.
- Nền tảng thực thi: Tensorflow 2.3.
- Mã nguồn: <https://github.com/hunglc007/tensorflow-yolov4-tflite>, mã nguồn này được tác giả của YOLO giới thiệu.

## II. CHỨC NĂNG CỦA HỆ THỐNG

### 1. Đổi với client

- Được xây dựng bằng html/css/js, với các chức năng:

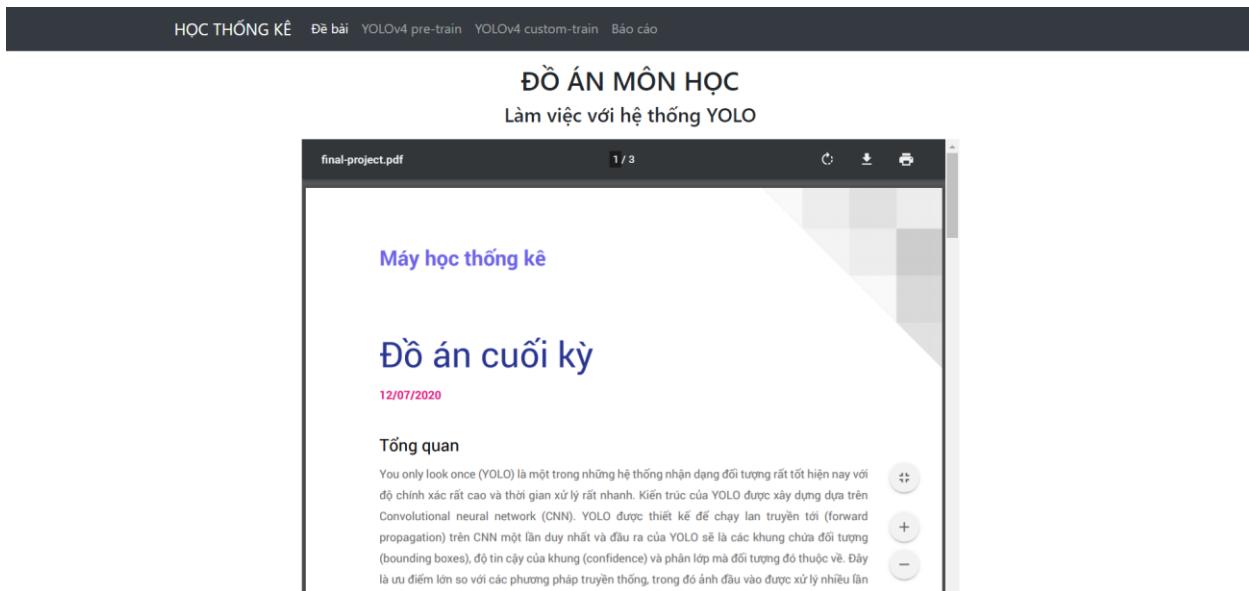
#### a. Hiển thị pdf

- Hiển thị đề bài và báo cáo để giảng viên và sinh viên tiện theo dõi. Sử dụng thẻ embed và đặt đường dẫn file pdf muốn hiển thị vào thuộc tính src, nếu muốn chiều dài và chiều rộng theo ý muốn, ta sử dụng thuộc tính width và height.

Code:

```
<embed src="../assets/pdf/final-project.pdf" width="800px" height="600px" />
```

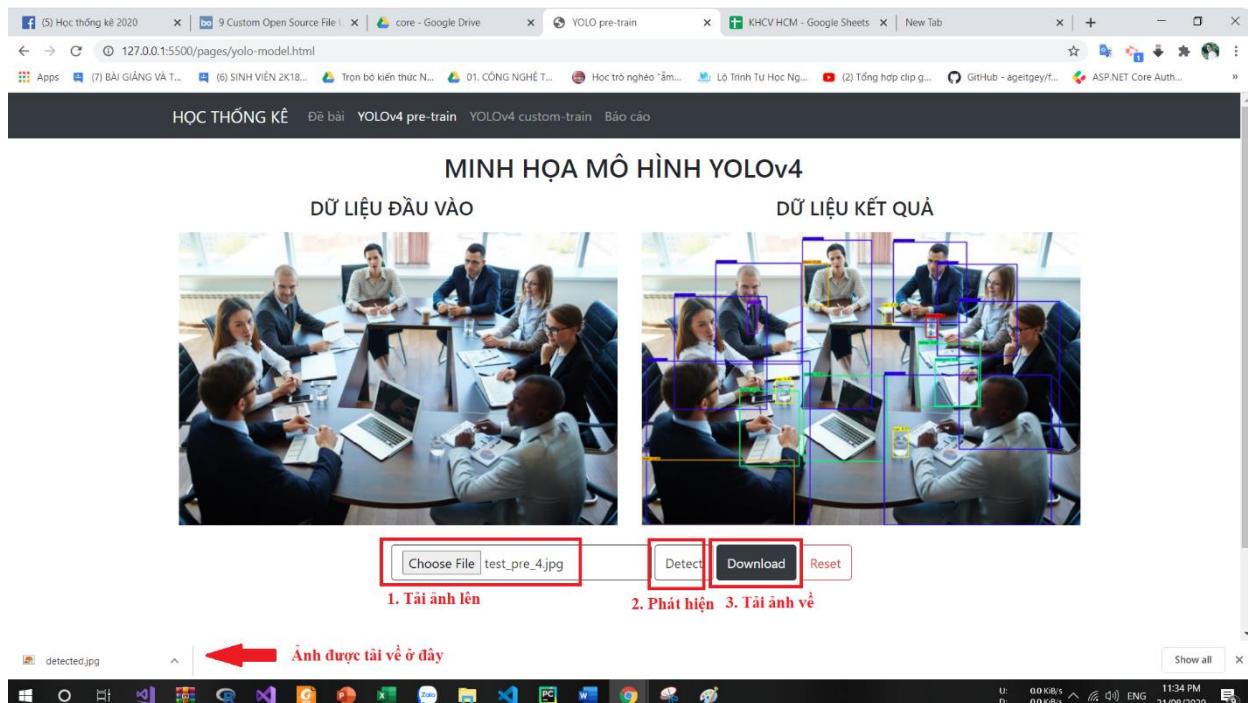
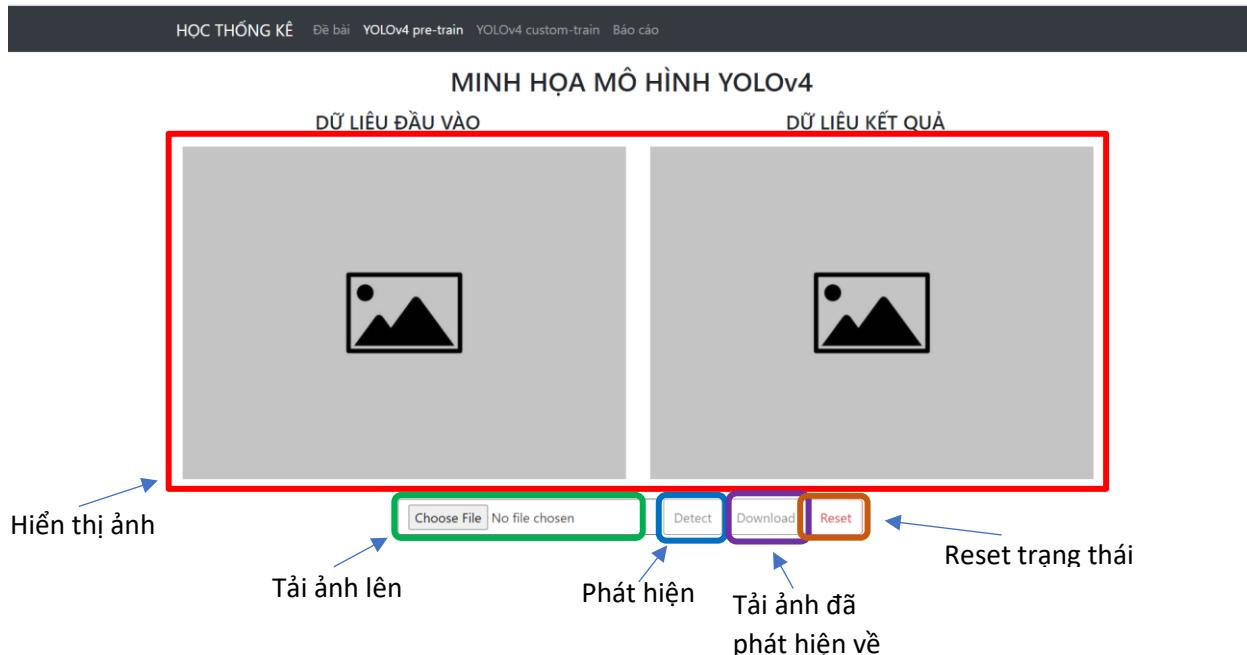
Hiển thị:



#### b. Phát hiện nhiều đối tượng trên ảnh

- Giúp người dùng đóng khung và gán nhãn được các đối tượng trên ảnh. Số lượng đối tượng được giới hạn trong tập dữ liệu COCO.
- Về mặt kỹ thuật: bao gồm các chức năng hiển thị ảnh, tải ảnh lên, tải ảnh xuống, phát hiện nhiều đối tượng trong ảnh và reset trạng thái.
  - Hiển thị ảnh: Mặc định ảnh hiển thị sẽ là ảnh trống.

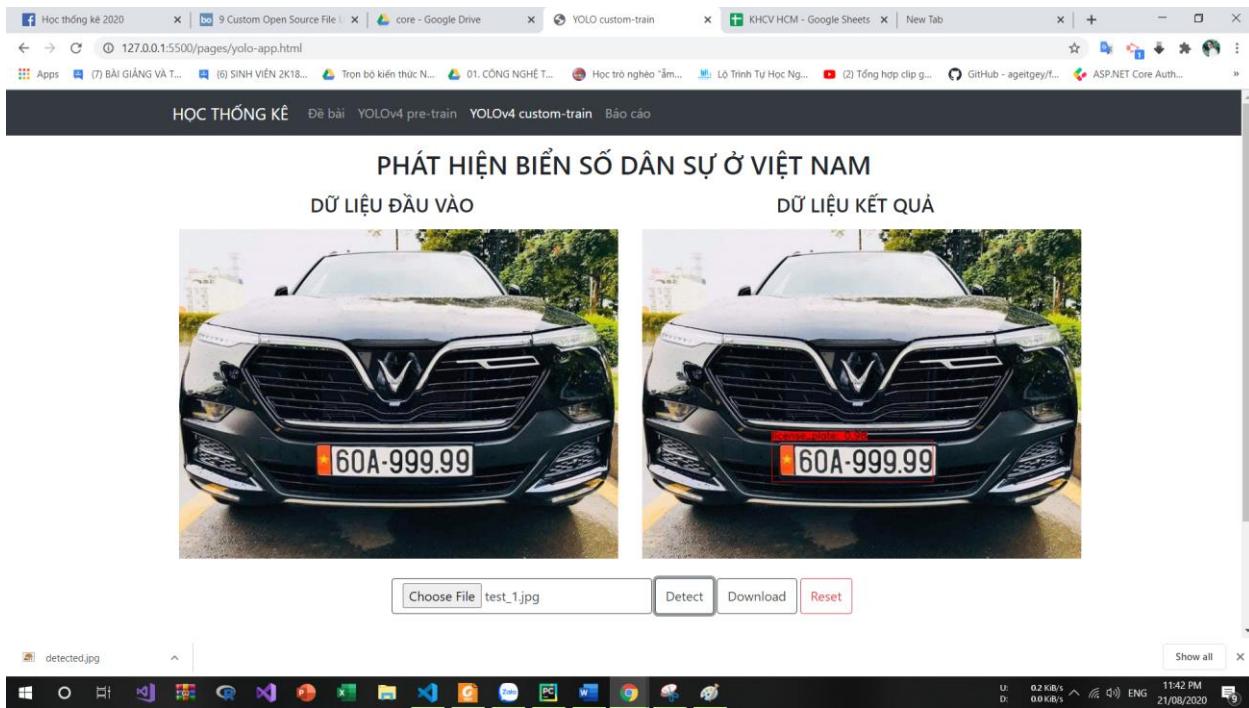
- Tải ảnh lên: Chức năng giúp người dùng có thể tải ảnh từ máy tính của mình lên trên hệ thống.
- Tải ảnh xuống: Chức năng giúp người dùng có thể tải ảnh sau khi thực hiện detect với độ phân giải như ảnh gốc ban đầu.
- Phát hiện nhiều đối tượng: Chức năng này giúp cho người dùng có thể gửi yêu cầu với dữ liệu hình ảnh cần detect cho server và nhận ảnh trả về để hiển thị hoặc tải xuống.
- Reset trạng thái: Đưa hệ thống về trạng thái ban đầu.



- Phần code: vui lòng tham khảo source code đính kèm.

### c. Phát hiện biển số xe dân sự

- Giúp cho người dùng đóng khung được vùng chứa biển số xe dân sự ở Việt Nam. Có thể sử dụng chức năng này như một công đoạn của việc nhận diện biển số xe dân sự ở Việt Nam, phạm vi áp dụng rộng rãi: hỗ trợ kiểm soát ở các bãi giữ xe, giúp hỗ trợ cảnh sát giao thông (phạt nguội) trong việc xử lý các phương tiện vi phạm, hỗ trợ trong việc tìm kiếm những chiếc xe bị mất trộm, thất lạc...
- Về mặt kỹ thuật: bao gồm các chức năng tương tự phát hiện nhiều đối tượng trên ảnh. Khác ở chỗ gửi request về cho server với api khác.



## 2. Đối với server

### a. Nhận yêu cầu từ client

- Chuyển đổi request từ client sang thành dữ liệu dạng hình ảnh dưới dạng numpy để xử lý.

```
# Lấy file ảnh người dùng upload lên
image = request.files["image"].read()
# Convert sang dạng array image
image_stream = io.BytesIO(image)
file_bytes = np.asarray(bytearray(image_stream.read()), dtype=np.uint8)
original_image = cv2.imdecode(file_bytes, cv2.IMREAD_COLOR)
original_image = cv2.cvtColor(original_image, cv2.COLOR_BGR2RGB)

input_size = FLAGS.size

image_data = cv2.resize(original_image, (input_size, input_size))
image_data = image_data / 255.

images_data = []
for i in range(1):
    images_data.append(image_data)
images_data = np.asarray(images_data).astype(np.float32)
```

b. Phát hiện đối tượng bằng YOLO

- Truyền ảnh đã được chuyển dưới dạng numpy để đưa vào mô hình YOLO với trọng số tùy chọn cho từng mục đích, mỗi trọng số sử dụng vào một api riêng biệt.
- Lưu ảnh đã được xử lý vào thư mục riêng biệt.
- *Phần này khá dài, vui lòng tham khảo trong file app.py của source code được cung cấp.*

c. Trả kết quả về cho client

- Đọc ảnh và chuyển đổi ảnh đã được xử lý qua YOLO sang dạng chuỗi base64 để gửi ảnh về server.

```
# convert image to byte type
with open(FLAGS.output, "rb") as image_file:
    encoded_string1 = base64.b64encode(image_file.read())

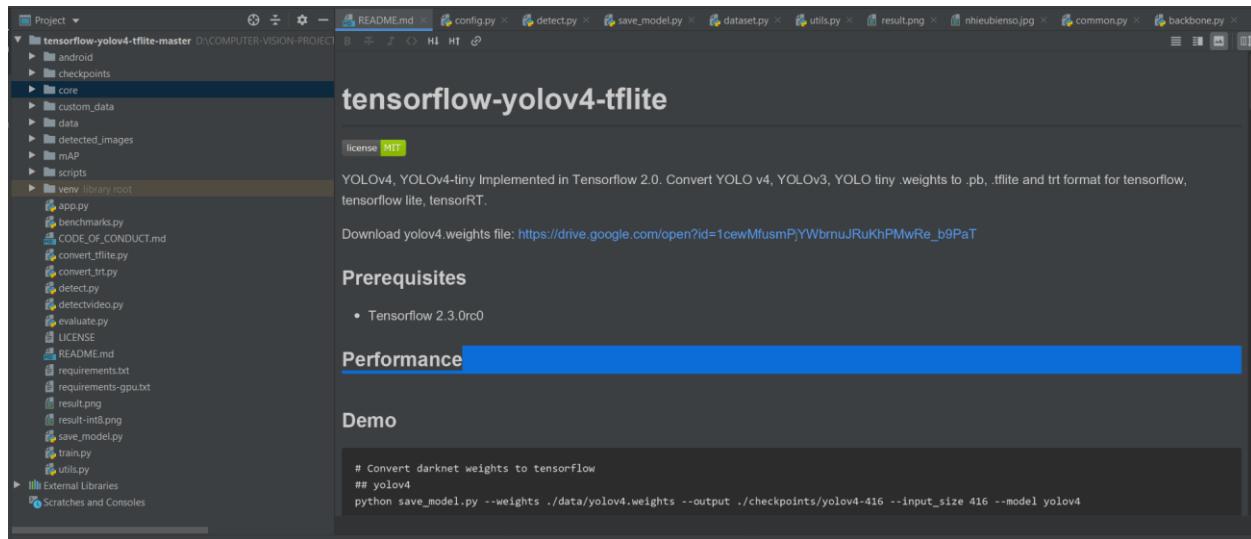
# convert byte to string
encoded_string = encoded_string1.decode("utf-8")
```

### III. CÀI ĐẶT YOLOv4 TRÊN TENSORFLOW

#### 1. Cài đặt trên PyCharm

Chúng ta sẽ cài đặt YOLOv4 dựa trên framework Tensorflow 2.3.0. Mã nguồn được chính tác giả giới thiệu. Có thể truy cập mã nguồn tại đây: <https://github.com/hunglc007/tensorflow-yolov4-tflite/>.

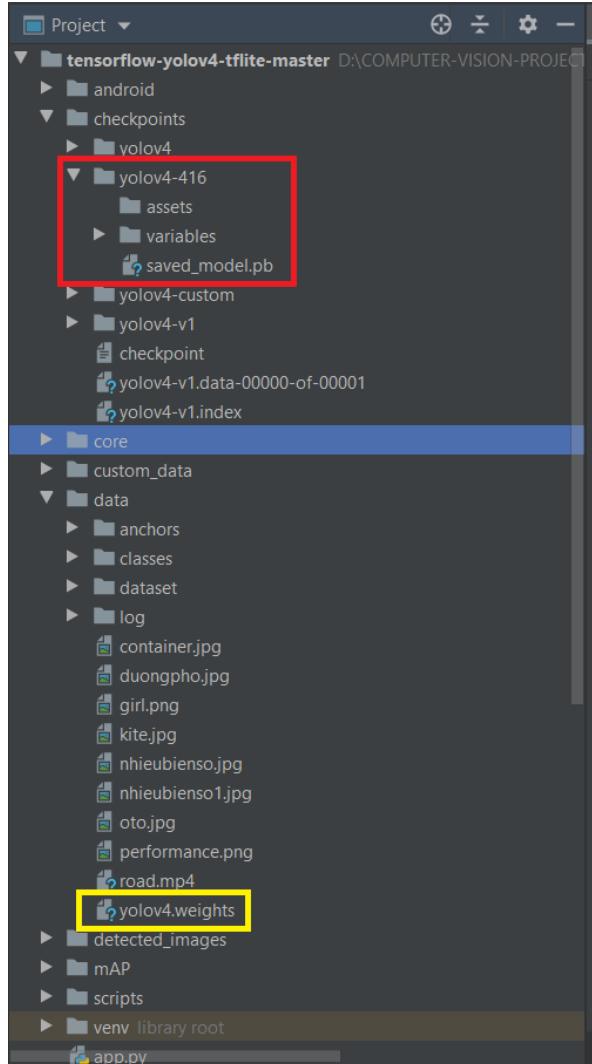
Có thể clone bằng git hoặc tải file source code dạng nén zip về máy.



Sau đó, cần phải cài đặt các package được yêu cầu trong file requirements.txt (đối với máy không có GPU) hoặc file requirements-gpu.txt (đối với máy có GPU).

Chúng ta sẽ sử dụng trọng số đã huấn luyện (pretrain) với tập dữ liệu COCO với 80 lớp, tải theo hướng dẫn trong file README.md. Vấn đề xảy ra ở đây là trọng số đó có dạng .weights được dùng cho mô hình gốc viết dưới C/C++, chúng ta cần chuyển sang dạng dùng cho Tensorflow bằng câu lệnh:

```
python save_model.py --weights ./data/yolov4.weights --output ./checkpoints/yolov4-416 --
input_size 416 --model yolov4
```

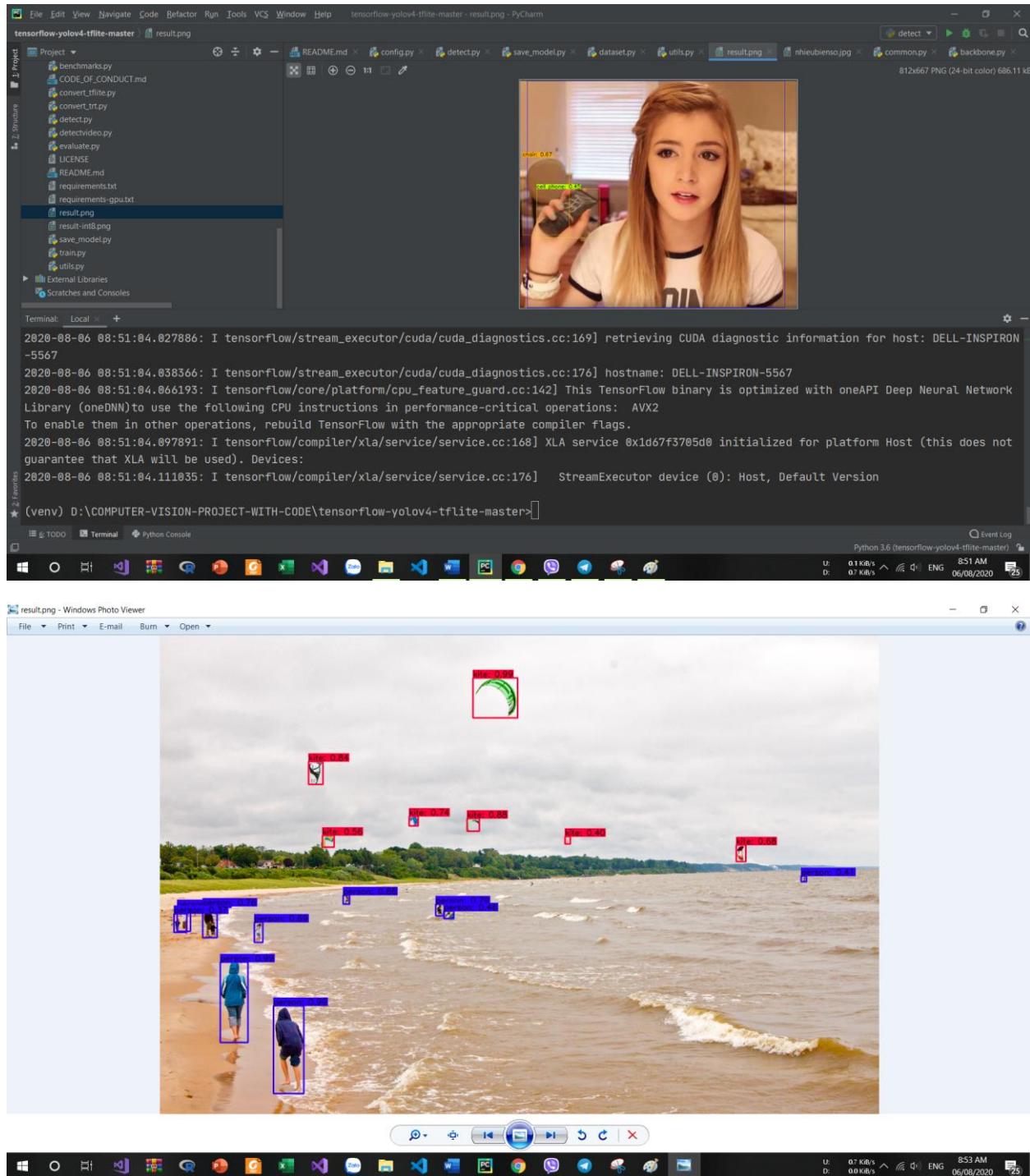


Ô vuông viền vàng: file trọng số dạng darknet, ô vuông viền đỏ: thư mục file trọng số dạng Tensorflow

Sau đó sử dụng bộ trọng số đã chuyển đổi trên để thử nghiệm trên một vài ảnh thông qua câu lệnh:

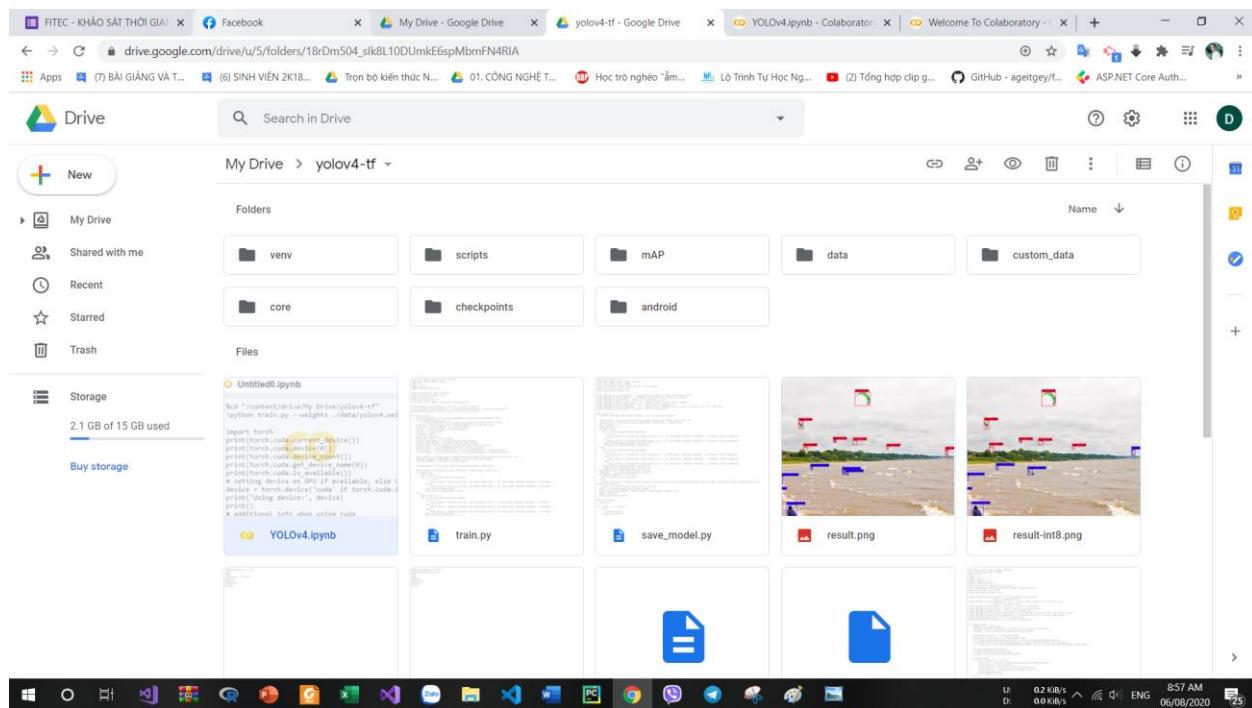
```
python detect.py --weights ./checkpoints/yolov4-416 --size 416 --model yolov4 --image ./data/kite.jpg
```

Một số kết quả:

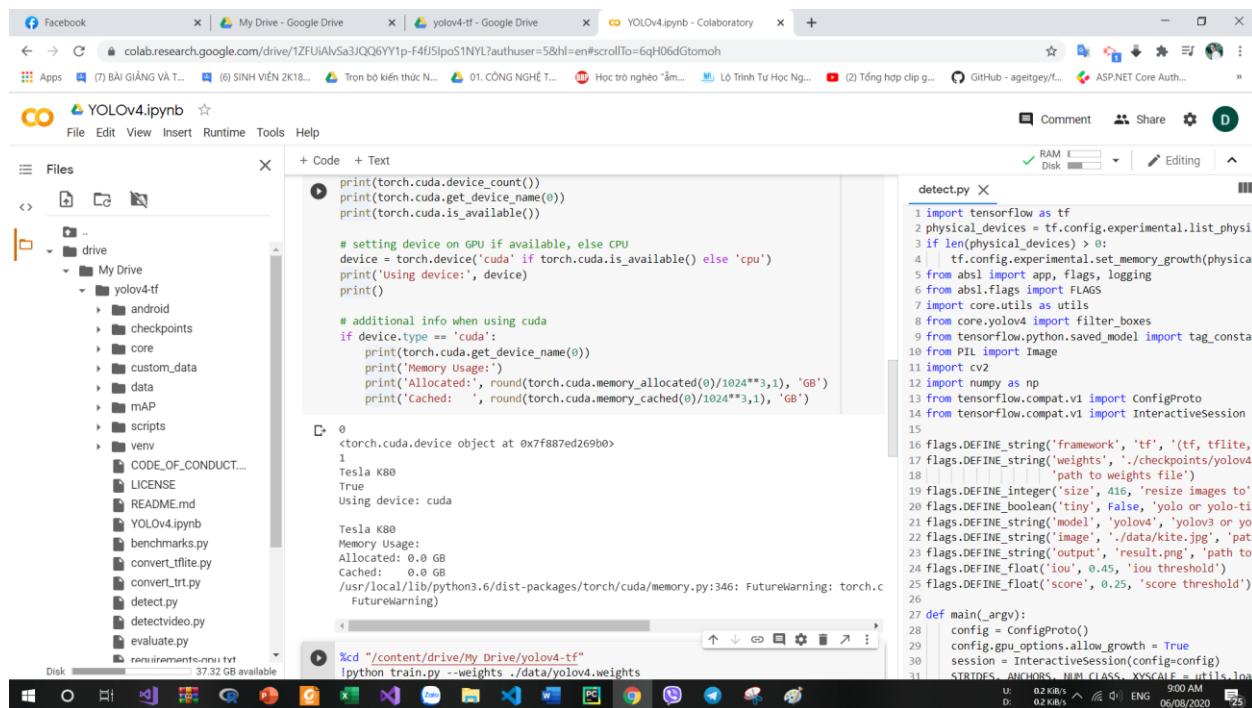


## 2. Cài đặt trên Google Colab

Chúng ta sẽ đẩy toàn bộ thư mục project của chúng ta lên Google Colab hoặc cài lại giống như PyCharm, thực hiện các lệnh trên cell của Google Colab.



Tạo một file YOLOv4.ipynb để thực hiện các câu lệnh trên đây.



Minh họa chạy trên cell của Google Colab.

```

Khoa học dữ liệu | Tổng hợp data - THI GIÁC M | Khóa: Học thống kê - 17_22 | yolov4-tf - Google Drive | YOLOv4.ipynb - Collaborator | Introduction - Bootstrap v4.5 | + | - | X
← → ⌂ colab.research.google.com/drive/1ZFUJAlvSa3jQQ6YYip-F4fJSipoS1NYL?authuser=5
Apps (7) BÀI GIẢNG VÀ T... (6) SINH VIÊN 2K18... Trợn bộ kiến thức N... 01. CÔNG NGHỆ T... Học trò nghèo "âm..." Lộ Trình Tự Học Ng... (2) Tổng hợp clip g... GitHub - ageitgey/f... ASP.NET Core Auth...
Commentaire Partager D
Fichier Modifier Affichage Insérer Exécution Outils Aide Dernier enregistrement effectué à 09:11
Fichiers + Code + Texte
Connexion à un environnement d'exécution pour permettre la navigation dans les fichiers.
%cd "/content/drive/My Drive/yolov4-tf"
!python train.py --weights ./data/yolov4.weights
Streaming output truncated to the last 5000 lines.
>> TEST STEP 1651 giou_loss: 0.38 conf_loss: 6.91 prob_loss: 0.00 total_loss: 7.30
>> TEST STEP 1651 giou_loss: 0.35 conf_loss: 6.91 prob_loss: 0.00 total_loss: 7.26
>> TEST STEP 1651 giou_loss: 0.36 conf_loss: 6.91 prob_loss: 0.00 total_loss: 7.21
>> TEST STEP 1651 giou_loss: 0.42 conf_loss: 6.93 prob_loss: 0.00 total_loss: 7.35
>> TEST STEP 1651 giou_loss: 0.30 conf_loss: 6.91 prob_loss: 0.00 total_loss: 7.21
>> TEST STEP 1651 giou_loss: 0.31 conf_loss: 6.91 prob_loss: 0.00 total_loss: 7.22
>> TEST STEP 1651 giou_loss: 0.36 conf_loss: 6.91 prob_loss: 0.00 total_loss: 7.27
>> TEST STEP 1651 giou_loss: 0.27 conf_loss: 6.92 prob_loss: 0.00 total_loss: 7.20
>> TEST STEP 1651 giou_loss: 0.33 conf_loss: 6.91 prob_loss: 0.00 total_loss: 7.24
>> TEST STEP 1651 giou_loss: 0.26 conf_loss: 6.90 prob_loss: 0.00 total_loss: 7.16
>> TEST STEP 1651 giou_loss: 0.45 conf_loss: 6.91 prob_loss: 0.00 total_loss: 7.36
>> TEST STEP 1651 giou_loss: 0.44 conf_loss: 6.91 prob_loss: 0.00 total_loss: 7.35
>> TEST STEP 1651 giou_loss: 0.26 conf_loss: 6.91 prob_loss: 0.00 total_loss: 7.17
>> TEST STEP 1651 giou_loss: 0.33 conf_loss: 6.91 prob_loss: 0.00 total_loss: 7.24
>> TEST STEP 1651 giou_loss: 0.42 conf_loss: 6.90 prob_loss: 0.00 total_loss: 7.32
>> TEST STEP 1651 giou_loss: 0.34 conf_loss: 6.92 prob_loss: 0.00 total_loss: 7.27
>> TEST STEP 1651 giou_loss: 0.37 conf_loss: 6.92 prob_loss: 0.00 total_loss: 7.30
>> TEST STEP 1651 giou_loss: 0.32 conf_loss: 6.91 prob_loss: 0.00 total_loss: 7.24
>> TEST STEP 1651 giou_loss: 0.39 conf_loss: 6.91 prob_loss: 0.00 total_loss: 7.30
>> TEST STEP 1651 giou_loss: 0.39 conf_loss: 6.91 prob_loss: 0.00 total_loss: 7.29
>> TEST STEP 1651 giou_loss: 0.38 conf_loss: 6.91 prob_loss: 0.00 total_loss: 7.28
>> TEST STEP 1651 giou_loss: 0.44 conf_loss: 6.91 prob_loss: 0.00 total_loss: 7.35
>> TEST STEP 1651 giou_loss: 0.36 conf_loss: 6.92 prob_loss: 0.00 total_loss: 7.27
>> TEST STEP 1651 giou_loss: 0.31 conf_loss: 6.92 prob_loss: 0.00 total_loss: 7.22
>> TEST STEP 1651 giou_loss: 0.36 conf_loss: 6.91 prob_loss: 0.00 total_loss: 7.27
>> TEST STEP 1651 giou_loss: 0.39 conf_loss: 6.91 prob_loss: 0.00 total_loss: 7.31
>> TEST STEP 1651 giou_loss: 0.35 conf_loss: 6.91 prob_loss: 0.00 total_loss: 7.26
>> TEST STEP 1651 giou_loss: 0.38 conf_loss: 6.91 prob_loss: 0.00 total_loss: 7.29
>> TEST STEP 1651 giou_loss: 0.37 conf_loss: 6.91 prob_loss: 0.00 total_loss: 7.28

```

U: 16 KB/s D: 20 KB/s ENG 11:22 AM 06/08/2020

```

Khoa học dữ liệu | Tổng hợp data - THI GIÁC M | Khóa: Học thống kê - 17_22 | yolov4-tf - Google Drive | YOLOv4.ipynb - Collaborator | Introduction - Bootstrap v4.5 | + | - | X
← → ⌂ colab.research.google.com/drive/1ZFUJAlvSa3jQQ6YYip-F4fJSipoS1NYL?authuser=5
Apps (7) BÀI GIẢNG VÀ T... (6) SINH VIÊN 2K18... Trợn bộ kiến thức N... 01. CÔNG NGHỆ T... Học trò nghèo "âm..." Lộ Trình Tự Học Ng... (2) Tổng hợp clip g... GitHub - ageitgey/f... ASP.NET Core Auth...
Commentaire Partager D
Fichier Modifier Affichage Insérer Exécution Outils Aide Dernier enregistrement effectué à 09:11
Fichiers + Code + Texte
RAM Disque Modification
%cd "/content/drive/My Drive/yolov4-tf"
!python save_model.py --weights ./data/yolov4.weights --output ./checkpoints/yolov4-416 --input_size 416 --model yolov4
/content/drive/My Drive/yolov4-tf
2020-08-06 02:08:25.677271: I tensorflow/stream_executor/platform/default/dso_loader.cc:48] Successfully opened dynamic library libcudart.so.10
2020-08-06 02:08:27.432642: I tensorflow/stream_executor/platform/default/dso_loader.cc:48] Successfully opened dynamic library libcuda.so.1
2020-08-06 02:08:27.436281: I tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:982] successful NUMA node read from SysFS had negative value
2020-08-06 02:08:27.437061: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1716] Found device 0 with properties:
pciBusID: 0000:00:04.0 name: Tesla K80 computeCapability: 3.7
coreClock: 0.8235GHz coreCount: 13 deviceMemorySize: 11.176GB deviceMemoryBandwidth: 223.96GiB/s
2020-08-06 02:08:27.437106: I tensorflow/stream_executor/platform/default/dso_loader.cc:48] Successfully opened dynamic library libcudart.so.10
2020-08-06 02:08:27.443292: I tensorflow/stream_executor/platform/default/dso_loader.cc:48] Successfully opened dynamic library libcublas.so.10
2020-08-06 02:08:27.445095: I tensorflow/stream_executor/platform/default/dso_loader.cc:48] Successfully opened dynamic library libcufft.so.10
2020-08-06 02:08:27.445455: I tensorflow/stream_executor/platform/default/dso_loader.cc:48] Successfully opened dynamic library libcurand.so.10
2020-08-06 02:08:27.448960: I tensorflow/stream_executor/platform/default/dso_loader.cc:48] Successfully opened dynamic library libcudnn.so.7
2020-08-06 02:08:27.453421: I tensorflow/stream_executor/platform/default/dso_loader.cc:48] Successfully opened dynamic library libcusparse.so.10
2020-08-06 02:08:27.459396: I tensorflow/stream_executor/platform/default/dso_loader.cc:48] Successfully opened dynamic library libdnn.so.7
2020-08-06 02:08:27.459559: I tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:982] successful NUMA node read from SysFS had negative value
2020-08-06 02:08:27.460456: I tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:982] successful NUMA node read from SysFS had negative value
2020-08-06 02:08:27.461198: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1858] Adding visible gpu devices: 0
2020-08-06 02:08:27.461609: I tensorflow/core/platform/cpu_feature_guard.cc:142] This TensorFlow binary is optimized with oneAPI Deep Neural Network enable them in other operations, rebuild TensorFlow with the appropriate compiler flags.
2020-08-06 02:08:27.467950: I tensorflow/core/platform/profile_utils/cu0_utils.cc:104] CPU Frequency: 2300000000 Hz
2020-08-06 02:08:27.468206: I tensorflow/compiler/xla/service/service.cc:1168] XLA service 0xf93400 initialized for platform Host (this does not support CUDA)
2020-08-06 02:08:27.468246: I tensorflow/compiler/xla/service/service.cc:1168] StreamExecutor device (0): Host, Default Version
2020-08-06 02:08:27.526975: I tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:982] successful NUMA node read from SysFS had negative value
2020-08-06 02:08:27.526996: I tensorflow/compiler/xla/service/service.cc:1168] XLA service 0xf93400 initialized for platform CUDA (this does not support Host)
2020-08-06 02:08:27.527035: I tensorflow/compiler/xla/service/service.cc:1168] StreamExecutor device (0): Tesla K80, Compute Capability 3.7
2020-08-06 02:08:27.527269: I tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:982] successful NUMA node read from SysFS had negative value
2020-08-06 02:08:27.528148: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1716] Found device 0 with properties:
pciBusID: 0000:00:04.0 name: Tesla K80 computeCapability: 3.7
coreClock: 0.8235GHz coreCount: 13 deviceMemorySize: 11.176GB deviceMemoryBandwidth: 223.96GiB/s
2020-08-06 02:08:27.528209: I tensorflow/stream_executor/platform/default/dso_loader.cc:48] Successfully opened dynamic library libcudart.so.10
2020-08-06 02:08:27.528307: I tensorflow/stream_executor/platform/default/dso_loader.cc:48] Successfully opened dynamic library libcublas.so.10

```

Disque 37.32 GB disponible(s) U: 02 KB/s D: 05 KB/s ENG 11:22 AM 06/08/2020

## IV. HUẤN LUYỆN YOLOv4 TRÊN TẬP DỮ LIỆU TÙY CHỈNH

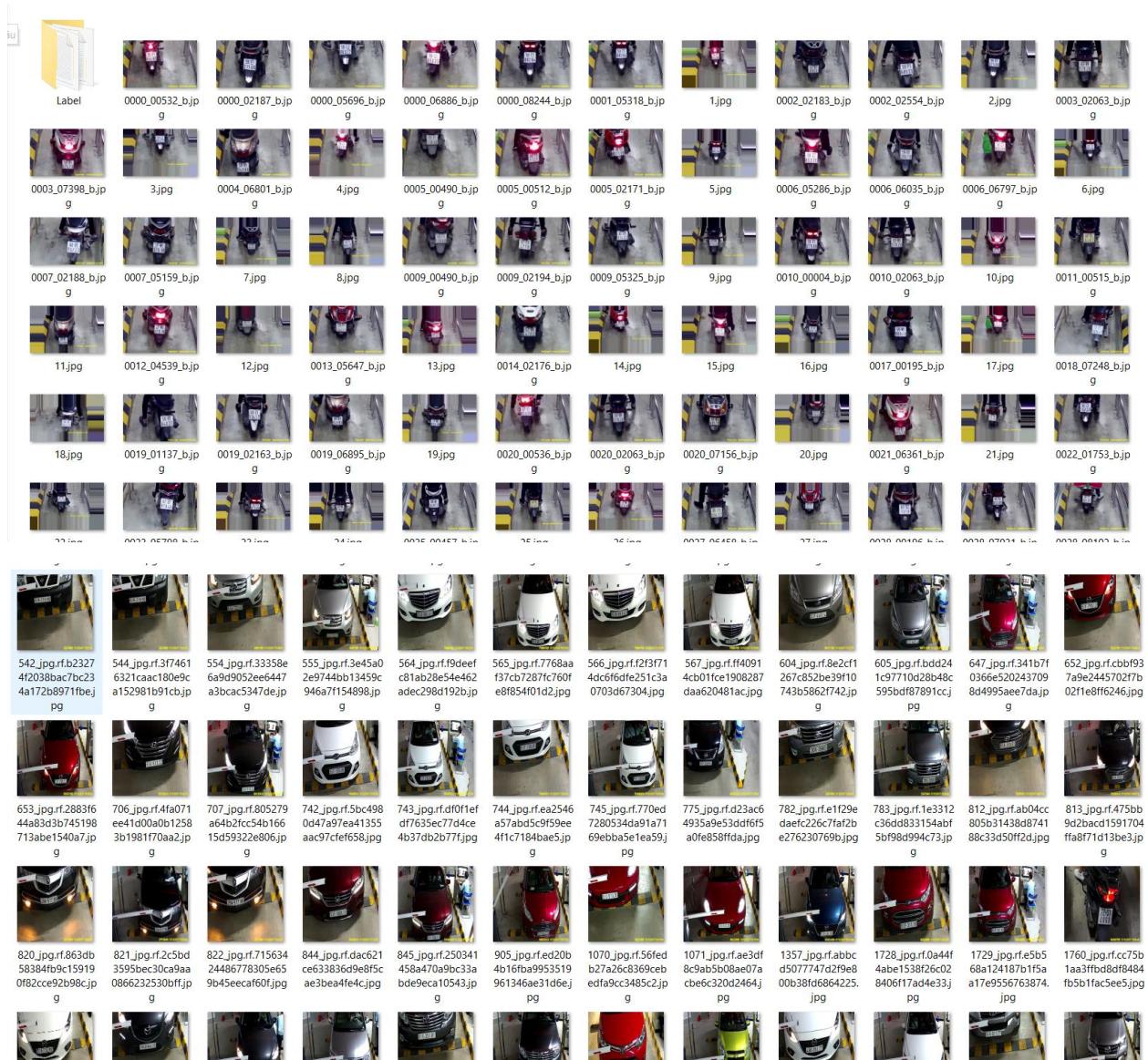
### 1. Bài toán

Phát hiện biển số xe dân sự ở Việt Nam.

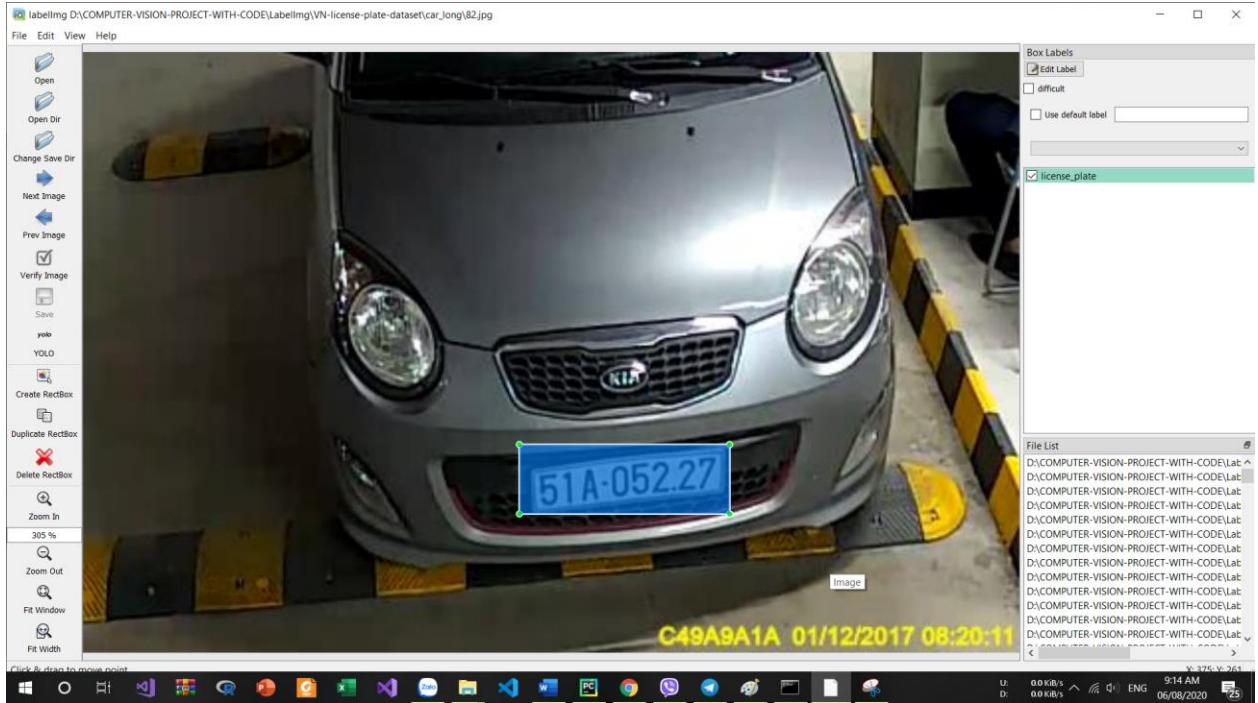
## 2. Chuẩn bị dữ liệu

Sử dụng tập dữ liệu có sẵn trên internet. Nguồn tham khảo: <https://thigiacmaytinh.com/tai-nguyen-xu-ly-anh/tong-hop-data-xu-ly-anh/>.

Mô tả dữ liệu: dữ liệu được trích xuất tại bãi quầy kiểm soát ra vào tại một bãi giữ xe, bao gồm biển số xe máy và biển số xe hơi.

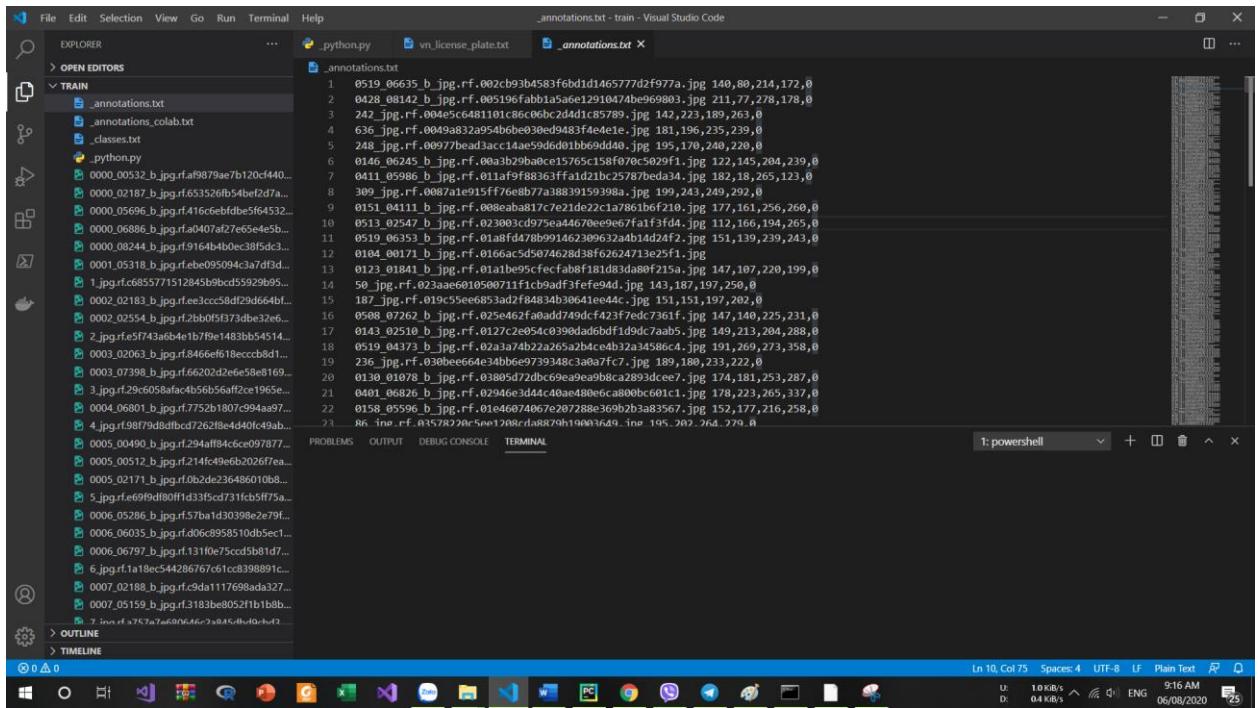


Sử dụng công cụ LabelImg để gán nhãn dữ liệu huấn luyện.



Sử dụng Roboflow để chuẩn hóa dữ liệu về dạng để huấn luyện cho mô hình YOLO.

Vì chúng ta sẽ huấn luyện trên Google Colab (để dùng GPU miễn phí được cung cấp bởi Google) nên chúng ta cần thay đổi một chút về đường dẫn đến ảnh.



Dữ liệu ban đầu khi chưa thêm đường dẫn

Thực thi một số dòng code thêm đường dẫn ở mỗi dòng:

```

python.py - train - Visual Studio Code
File Edit Selection View Go Run Terminal Help
python.py
vn_license_plate.txt
_annotations.txt

1 newf=""
2 with open('_annotations.txt','r') as f:
3     for line in f:
4         newf+="/content/drive/My Drive/yolov4-tf/custom_data/dataset/" + line.strip()+"\n"
5     # newf += .../custom_data/dataset/' + line.strip()+'\n'
6     f.close()
7 with open('vn_license_plate.txt','w') as f:
8     f.write(newf)
9     f.close()

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

1: powershell

Python 3.8.3 64-bit ('Vietnam-License-Plate-Recognition': conda) 0 Δ 0 Python extension loading...

Ln 9, Col 14 Spaces: 4 UTF-8 CRLF Python Go Live

U: 0.0 KB/s D: 0.0 KB/s ENG 9:17 AM 06/08/2020 25

Và kết quả là:

```

File Edit Selection View Go Run Terminal Help
vn_license_plate.txt - train - Visual Studio Code
python.py
vn_license_plate.txt
_annotations.txt

1 /content/drive/My Drive/yolov4-tf/custom_data/dataset/0519_06635_b.jpg,rf.002cb93fb583bfbd1d465777d2f977a.jpg 140,80,214,172,0
2 /content/drive/My Drive/yolov4-tf/custom_data/dataset/0428_08142_b.jpg,rf.005196fab1a5a6e1291047ab9e699803.jpg 211,77,278,178,0
3 /content/drive/My Drive/yolov4-tf/custom_data/dataset/242.jpg,rf.004e5c481101c86c06b2d41c85789.jpg 142,223,189,263,0
4 /content/drive/My Drive/yolov4-tf/custom_data/dataset/636.jpg,rf.0049a832a95b68be030ed9483fa4e1e.jpg 181,196,235,239,0
5 /content/drive/My Drive/yolov4-tf/custom_data/dataset/248.jpg,rf.00977beadacc1ca4e59dd01bb69d40.jpg 195,170,240,220,0
6 /content/drive/My Drive/yolov4-tf/custom_data/dataset/0146_06245_b.jpg,rf.003b29ba0cc01576c158f070c5920f1.jpg 122,145,204,239,0
7 /content/drive/My Drive/yolov4-tf/custom_data/dataset/0411_09586_b.jpg,rf.011af9188363ffaf2d1bc25787beda34.jpg 182,18,265,123,0
8 /content/drive/My Drive/yolov4-tf/custom_data/dataset/309.jpg,rf.00871e0915f7fe6b72a8839193989a.jpg 199,243,249,292,0
9 /content/drive/My Drive/yolov4-tf/custom_data/dataset/0151_04111_b.jpg,rf.008ea0ba817c7e2d2c1a7861bf210.jpg 177,161,256,260,0
10 /content/drive/My Drive/yolov4-tf/custom_data/dataset/0513_02547_b.jpg,rf.023003cd975ea446700ee967a1f3fd4.jpg 112,166,194,265,0
11 /content/drive/My Drive/yolov4-tf/custom_data/dataset/0519_06353_b.jpg,rf.01a8f4d7889914623096324d4d1d24f2.jpg 151,139,239,243,0
12 /content/drive/My Drive/yolov4-tf/custom_data/dataset/0104_00171_b.jpg,rf.0166ac5d9074628038f62624713e2f1.jpg
13 /content/drive/My Drive/yolov4-tf/custom_data/dataset/0123_01841_b.jpg,rf.01a1b95cfcabef181d83d4080f215a.jpg 147,107,220,199,0
14 /content/drive/My Drive/yolov4-tf/custom_data/dataset/50.jpg,rf.023aae6010508711fc1bc9adff3ffef94d.jpg 143,187,197,250,0
15 /content/drive/My Drive/yolov4-tf/custom_data/dataset/187.jpg,rf.019c5e66853d2f84834b30641ee44c.jpg 151,151,197,202,0
16 /content/drive/My Drive/yolov4-tf/custom_data/dataset/0508_07262_b.jpg,rf.025e462fa0add749dcfa23f7edc7361f.jpg 147,140,225,231,0
17 /content/drive/My Drive/yolov4-tf/custom_data/dataset/0143_02518_b.jpg,rf.0127c2e054c3909ad6bfdf9d9dc7aab5.jpg 149,213,204,288,0
18 /content/drive/My Drive/yolov4-tf/custom_data/dataset/0519_04373_b.jpg,rf.02a3a7b2a265a2b4cedb12a3d5806c4.jpg 191,269,273,358,0
19 /content/drive/My Drive/yolov4-tf/custom_data/dataset/233.jpg,rf.030be066a34b66973943c3aa07fc7.jpg 189,188,233,222,0
20 /content/drive/My Drive/yolov4-tf/custom_data/dataset/0130_01078_b.jpg,rf.028805d2d4bc69a9ea98c3a2893ddee7.jpg 174,181,253,287,0
21 /content/drive/My Drive/yolov4-tf/custom_data/dataset/0401_06826_b.jpg,rf.02946e3d4c404e088e6ca800bc601c1.jpg 178,223,265,337,0
22 /content/drive/My Drive/yolov4-tf/custom_data/dataset/0158_05596_b.jpg,rf.01e46004067888e659f2b3a83567.jpg 152,177,216,258,0
23 /content/drive/My Drive/yolov4-tf/custom_data/dataset/186.ina,rf.04578772dc5ee120887cf1900f6d91na.105,397,34d,79,0

vn_license_plate.txt

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL


1: powershell



Python 3.8.3 64-bit ('Vietnam-License-Plate-Recognition': conda) 0 Δ 0 Python extension loading...



Ln 1, Col 1 Spaces: 4 UTF-8 CRLF Plain Text Go Live



U: 0.1 KB/s D: 14 KB/s ENG 9:17 AM 06/08/2020 25


```

Lưu ý: tùy vào việc mounting vào Colab và tổ chức thư mục cho project trên Google Drive, chúng ta có những đường dẫn khác nhau để thêm vào.

Như vậy, chúng ta sẽ có các file chứa danh sách class \_classes.txt, file chứa tọa độ các bounding box từ các ảnh trong dataset vn\_license\_plate.txt và các file ảnh. Để thuận tiện chúng ta nên đổi tên \_classes.txt

thành vn\_license\_plate\_classes.txt và như mục của chúng ta bao gồm các file ảnh, file vn\_license\_plate\_classes.txt và file vn\_license\_plate.txt.

### 3. Tiến hành huấn luyện

Tải thư mục chúng ta đã chuẩn bị cho dữ liệu lên project trên Google Colab. Quá trình này diễn ra hơi mất nhiều thời gian, do số lượng ảnh của chúng ta là 3490 ảnh (nhiều ảnh kích thước nhỏ nên mất thời gian hơn ít ảnh kích thước lớn).

Sửa đổi file cfg trong thư mục core. Lưu ý các dòng

```
__C.YOLO.CLASSES = "./custom_data/classes/vn_license_plate_classes.txt",
__C.TRAIN.ANNOT_PATH = "./custom_data/dataset/vn_license_plate.txt",
__C.TEST.ANNOT_PATH = "./custom_data/dataset/vn_license_plate.txt",
__C.TEST.BATCH_SIZE = 10
```

để chúng ta thay đổi giá trị tương ứng với tập dữ liệu và giá trị batch size của chúng ta.

Thực hiện train (transfer learning) bằng câu lệnh:

```
python train.py --weights ./data/yolov4.weights
```

```
%cd "/content/drive/My Drive/yolov4-tf"
!python train.py --weights ./data/yolov4.weights

=> STEP 2951/16500 lr: 0.000949 giou_loss: 0.39 conf_loss: 1.97 prob_loss: 0.00 total_loss: 2.35
=> STEP 2952/16500 lr: 0.000949 giou_loss: 0.34 conf_loss: 1.96 prob_loss: 0.00 total_loss: 2.30
=> STEP 2953/16500 lr: 0.000949 giou_loss: 0.32 conf_loss: 1.96 prob_loss: 0.00 total_loss: 2.28
=> STEP 2954/16500 lr: 0.000949 giou_loss: 0.37 conf_loss: 1.96 prob_loss: 0.00 total_loss: 2.33
=> STEP 2955/16500 lr: 0.000949 giou_loss: 0.58 conf_loss: 2.02 prob_loss: 0.00 total_loss: 2.60
=> STEP 2956/16500 lr: 0.000949 giou_loss: 0.36 conf_loss: 1.96 prob_loss: 0.00 total_loss: 2.32
=> STEP 2957/16500 lr: 0.000949 giou_loss: 0.52 conf_loss: 2.07 prob_loss: 0.00 total_loss: 2.59
=> STEP 2958/16500 lr: 0.000949 giou_loss: 0.37 conf_loss: 1.95 prob_loss: 0.00 total_loss: 2.32
=> STEP 2959/16500 lr: 0.000949 giou_loss: 0.50 conf_loss: 2.01 prob_loss: 0.00 total_loss: 2.50
=> STEP 2960/16500 lr: 0.000949 giou_loss: 0.54 conf_loss: 2.05 prob_loss: 0.00 total_loss: 2.59
=> STEP 2961/16500 lr: 0.000949 giou_loss: 0.40 conf_loss: 1.95 prob_loss: 0.00 total_loss: 2.35
=> STEP 2962/16500 lr: 0.000949 giou_loss: 0.70 conf_loss: 2.12 prob_loss: 0.00 total_loss: 2.82
=> STEP 2963/16500 lr: 0.000949 giou_loss: 0.50 conf_loss: 2.02 prob_loss: 0.00 total_loss: 2.53
=> STEP 2964/16500 lr: 0.000949 giou_loss: 0.38 conf_loss: 1.95 prob_loss: 0.00 total_loss: 2.32
=> STEP 2965/16500 lr: 0.000949 giou_loss: 0.37 conf_loss: 1.95 prob_loss: 0.00 total_loss: 2.31
=> STEP 2966/16500 lr: 0.000949 giou_loss: 0.32 conf_loss: 1.94 prob_loss: 0.00 total_loss: 2.26
=> STEP 2967/16500 lr: 0.000949 giou_loss: 0.37 conf_loss: 1.94 prob_loss: 0.00 total_loss: 2.31
=> STEP 2968/16500 lr: 0.000949 giou_loss: 0.33 conf_loss: 1.94 prob_loss: 0.00 total_loss: 2.27
=> STEP 2969/16500 lr: 0.000949 giou_loss: 0.72 conf_loss: 2.11 prob_loss: 0.00 total_loss: 2.83
=> STEP 2970/16500 lr: 0.000948 giou_loss: 0.37 conf_loss: 1.94 prob_loss: 0.00 total_loss: 2.31
=> TEST STEP 2971 giou_loss: 0.40 conf_loss: 1.93 prob_loss: 0.00 total_loss: 2.34
=> TEST STEP 2971 giou_loss: 0.41 conf_loss: 1.94 prob_loss: 0.00 total_loss: 2.35
=> TEST STEP 2971 giou_loss: 0.46 conf_loss: 1.93 prob_loss: 0.00 total_loss: 2.39
=> TEST STEP 2971 giou_loss: 0.33 conf_loss: 1.94 prob_loss: 0.00 total_loss: 2.26
=> TEST STEP 2971 giou_loss: 0.37 conf_loss: 1.93 prob_loss: 0.00 total_loss: 2.30
=> TEST STEP 2971 giou_loss: 0.32 conf_loss: 1.93 prob_loss: 0.00 total_loss: 2.26
=> TEST STEP 2971 giou_loss: 0.44 conf_loss: 1.93 prob_loss: 0.00 total_loss: 2.37
=> TEST STEP 2971 giou_loss: 0.40 conf_loss: 1.94 prob_loss: 0.00 total_loss: 2.34
=> TEST STEP 2971 giou_loss: 0.29 conf_loss: 1.93 prob_loss: 0.00 total_loss: 2.22
=> TEST STEP 2971 giou_loss: 0.37 conf_loss: 1.94 prob_loss: 0.00 total_loss: 2.31
```

Tại epoch thứ 10, chúng ta tạm dừng chương trình và thu được các file: checkpoint, .yolov4.data-00001-of-00000, yolov4.index.

Vì chúng ta sẽ viết backend bằng Flask nên đem 3 file này về thư mục checkpoint tại project mở bằng PyCharm.

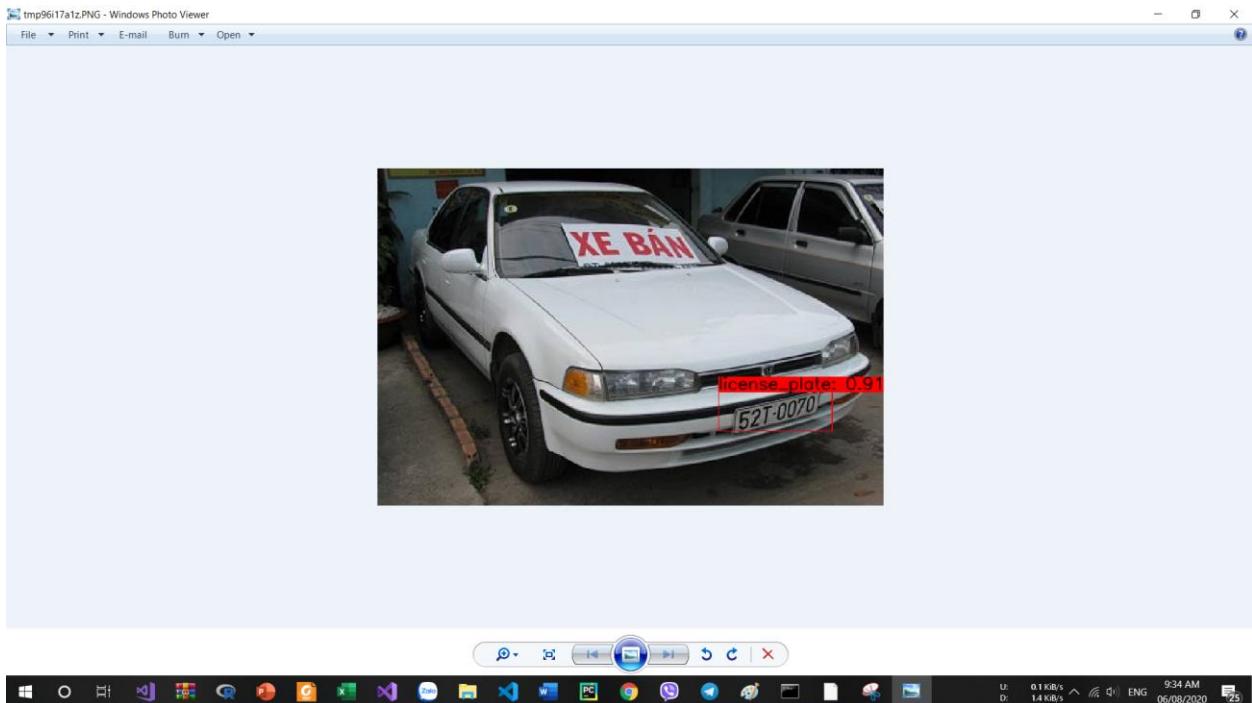
#### 4. Sử dụng trọng số huấn luyện

Sau khi train thành công, chúng ta sẽ thu được 3 file: checkpoint, yolov4.data-00001-of-00000 và yolov4.index. Để sử dụng được ba file này, chúng ta cần phải chỉnh lại file config.py như ta đã thực hiện trên Colab và thực hiện dòng lệnh:

```
python save_model.py --weights ./checkpoints/yolov4 --output ./checkpoints/yolov4-custom --  
input_size 416 --model yolov4
```

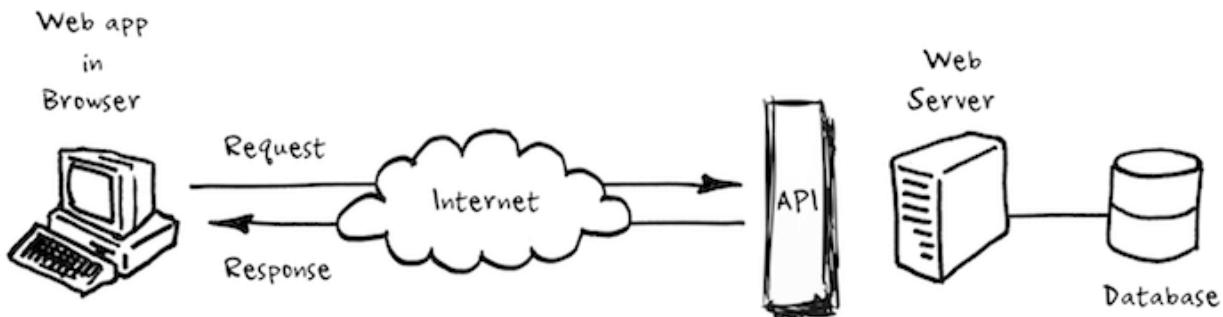
Sau đó, sử dụng trọng số đã chuyển đổi và quan sát kết quả thông qua câu lệnh:

```
python detect.py --weights ./checkpoints/yolov4-custom --size 416 --model yolov4 --image  
./data/oto.jpg.
```



Kết quả cho thấy, hệ thống đã nhận diện chính xác vùng có biển số xe. Như vậy, chúng ta đã huấn luyện thành công cho tập dữ liệu biển số xe dân sự Việt Nam, ở đây chúng ta không quan trọng hiệu suất phải đạt tối ưu và cao nhất do hạn chế về dữ liệu và thời gian huấn luyện trên Colab là hạn chế.

## V. XÂY DỰNG FLASK API CHO MÔ HÌNH



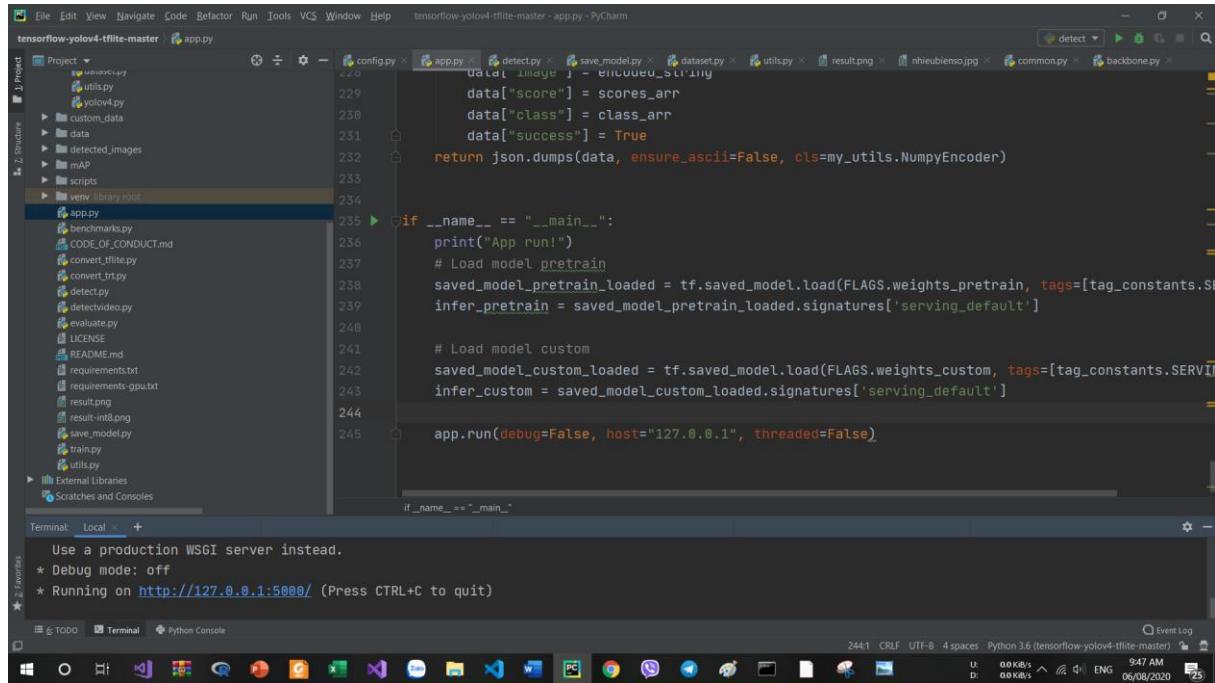
Mô hình client-server

Giới thiệu đôi chút về Flask, đây là một framework giúp chúng ta xây dựng các API dưới nền tảng python. Với nhu cầu dùng để xây dựng server cho hệ thống phát hiện đối tượng trong phạm vi đồ án thì Flask là đủ.

Trước tiên, chúng ta cần phải cài đặt package flask, flask\_cors, base64, json..., kèm theo đó là các hàm bổ sung trong file utils.py. Sau đó chúng ta sẽ kết hợp file detect.py để tạo API. Mã nguồn tham khảo tại đây.

Một số chỉnh sửa:

- Thay vì chúng ta sử dụng flags từ absl package để xử lý tham số dòng lệnh thì chúng ta sẽ thay thế bằng việc chuyển nó thành class.
- Sử dụng template đơn giản từ Flask và kết hợp file detect, lưu ý bước tải trọng số vào mô hình phải đặt trong hàm main để server chỉ cần khởi tạo nó khi hoạt động.



```

File Edit View Navigate Code Refactor Run Tools VCS Window Help tensorflow-yolov4-tflite-master - app.py - PyCharm
tensorflow-yolov4-tflite-master > app.py
Project config.py app.py detect.py save_model.py dataset.py utils.py result.png nhieubienso.jpg common.py backbone.py
utils.py
yolo4.py
custom_data
data
detected_images
mAP
scripts
venv library root
app.py
benchmarks.py
CODE_OF_CONDUCT.md
convert_tflite.py
convert_trt.py
detect.py
detectvideo.py
evaluate.py
LICENSE
README.md
requirements.txt
requirements-gpu.txt
result.png
result-int8.png
save_model.py
train.py
utils.py
External Libraries
Scratches and Consoles
Terminal Local +
Use a production WSGI server instead.
* Debug mode: off
* Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
Event Log
2441 CRLF UTF-8 4 spaces Python 3.6 (tensorflow-yolov4-tflite-master) 9:47 AM U: 0.0KB/s D: 0.0KB/s ENG 06/08/2020 25

```

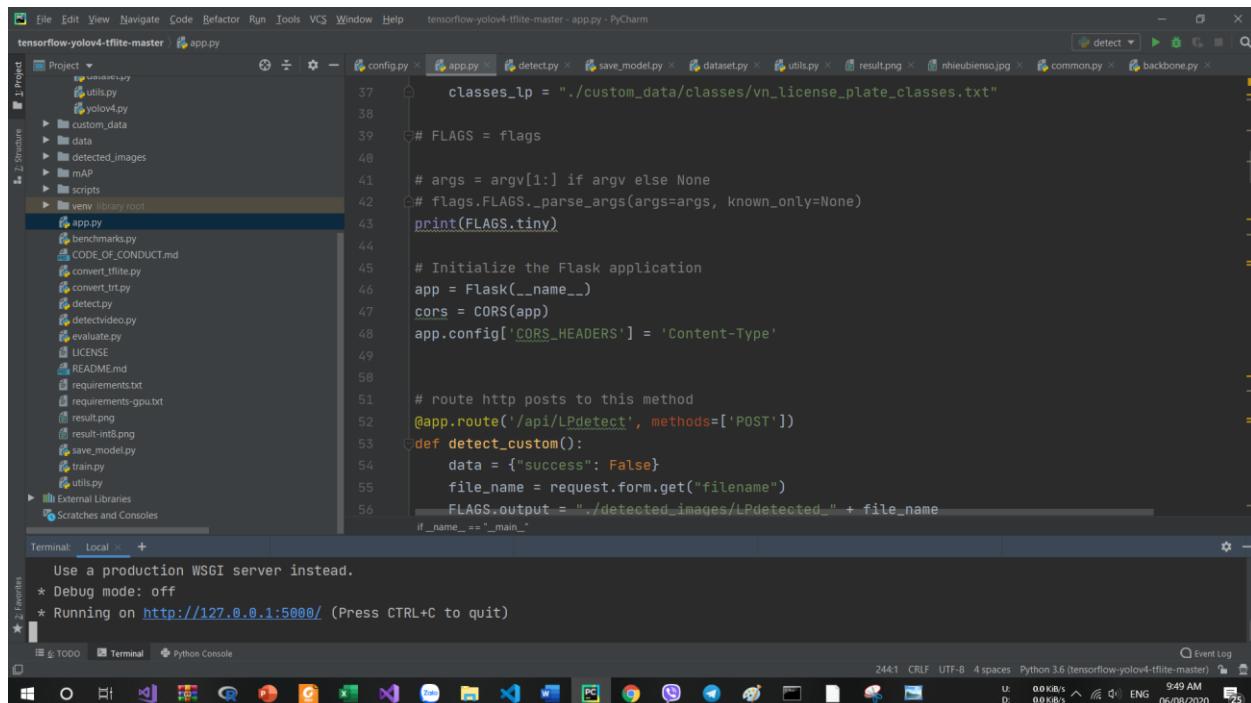
```

config.py
229     data["image"] = encode_base64(image)
230     data["score"] = scores_arr
231     data["class"] = class_arr
232     data["success"] = True
233
234     return json.dumps(data, ensure_ascii=False, cls=my_utils.NumpyEncoder)
235
236 if __name__ == "__main__":
237     print("App run!")
238     # Load model pretrain
239     saved_model_pretrain_loaded = tf.saved_model.load(FLAGS.weights_pretrain, tags=[tag_constants.SERVING])
240     infer_pretrain = saved_model_pretrain_loaded.signatures['serving_default']
241
242     # Load model custom
243     saved_model_custom_loaded = tf.saved_model.load(FLAGS.weights_custom, tags=[tag_constants.SERVING])
244     infer_custom = saved_model_custom_loaded.signatures['serving_default']
245
246     app.run(debug=False, host="127.0.0.1", threaded=False)
247
if __name__ == "__main__":

```

Một số điểm quan trọng khi tạo Flask API:

- Khai báo CORS để có thể gửi nhận tài nguyên từ các nguồn domain khác nhau.



```

File Edit View Navigate Code Refactor Run Tools VCS Window Help tensorflow-yolov4-tflite-master - app.py - PyCharm
tensorflow-yolov4-tflite-master > app.py
Project config.py app.py detect.py save_model.py dataset.py utils.py result.png nhieubienso.jpg common.py backbone.py
utils.py
yolo4.py
custom_data
data
detected_images
mAP
scripts
venv library root
app.py
benchmarks.py
CODE_OF_CONDUCT.md
convert_tflite.py
convert_trt.py
detect.py
detectvideo.py
evaluate.py
LICENSE
README.md
requirements.txt
requirements-gpu.txt
result.png
result-int8.png
save_model.py
train.py
utils.py
External Libraries
Scratches and Consoles
Terminal Local +
Use a production WSGI server instead.
* Debug mode: off
* Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
Event Log
2441 CRLF UTF-8 4 spaces Python 3.6 (tensorflow-yolov4-tflite-master) 9:49 AM U: 0.0KB/s D: 0.0KB/s ENG 06/08/2020 25

```

```

config.py
37     classes_lp = "./custom_data/classes/vn_license_plate_classes.txt"
38
39     # FLAGS = flags
40
41     # args = argv[1:] if argv else None
42     # flags.FLAGS._parse_args(args=args, known_only=None)
43     print(FLAGS.tiny)
44
45     # Initialize the Flask application
46     app = Flask(__name__)
47     cors = CORS(app)
48     app.config['CORS_HEADERS'] = 'Content-Type'
49
50     # route http posts to this method
51     @app.route('/api/LPdetect', methods=['POST'])
52     def detect_custom():
53         data = {"success": False}
54         file_name = request.form.get("filename")
55         FLAGS.output = "./detected_images/LPdetected_" + file_name
56
if __name__ == "__main__":

```

- Chuyển file ảnh nhận được từ client thành dạng ảnh trong opencv.

```

File Edit View Navigate Code Refactor Run Tools VCS Window Help tensorflow-yolov4-tflite-master - app.py - PyCharm
tensorflow-yolov4-tflite-master / app.py
Project: tensorflow-yolov4-tflite-master
  - app.py
  - config.py
  - detect.py
  - save_model.py
  - dataset.py
  - utils.py
  - result.png
  - nhieubienso.jpg
  - common.py
  - backbone.py
  - LICENSE
  - README.md
  - requirements.txt
  - requirements-gpu.txt
  - resultng
  - result-int8.png
  - save_modelpy
  - train.py
  - utils.py
  - External Libraries
  - Scratches and Consoles
  - library root
    - app.py
    - benchmarks.py
    - CODE_OF_CONDUCT.md
    - convert_tflite.py
    - convert_trt.py
    - detect.py
    - detectvideo.py
    - evaluate.py
    - LICENSE
    - README.md
    - requirements.txt
    - requirements-gpu.txt
    - resultng
    - result-int8.png
    - save_modelpy
    - train.py
    - utils.py
  - venv
    - libra
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
# route http posts to this method
@app.route('/api/LPdetect', methods=['POST'])
def detect_custom():
    data = {"success": False}
    file_name = request.form.get("filename")
    FLAGS.output = "./detected_images/LPdetected_" + file_name
    if request.files.get("image"):
        # Lấy file ảnh người dùng upload lên
        image = request.files["image"].read()
        # Convert sang dạng array image
        image_stream = io.BytesIO(image)
        file_bytes = np.asarray(bytearray(image_stream.read()), dtype=np.uint8)
        original_image = cv2.imdecode(file_bytes, cv2.IMREAD_COLOR)
        original_image = cv2.cvtColor(original_image, cv2.COLOR_BGR2RGB)

        input_size = FLAGS.size
        if __name__ == "__main__":
            image_data =
                app
                class FLAGS
                : input_size, input_size))
Use a production WSGI server instead.
* Debug mode: off
* Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
  - TODO
  - Terminal
  - Python Console
Event Log
2441 CRLF 4 spaces Python 3.6 (tensorflow-yolov4-tflite-master)
U: 0.0 KB/s D: 0.0 KB/s ENG 9:49 AM 06/08/2020

```

- Chuyển đổi ảnh từ opencv sang dạng chuỗi nhị phân base64.

```

File Edit View Navigate Code Refactor Run Tools VCS Window Help tensorflow-yolov4-tflite-master - app.py - PyCharm
tensorflow-yolov4-tflite-master / app.py
Project: tensorflow-yolov4-tflite-master
  - app.py
  - config.py
  - detect.py
  - save_model.py
  - dataset.py
  - utils.py
  - result.png
  - nhieubienso.jpg
  - common.py
  - backbone.py
  - LICENSE
  - README.md
  - requirements.txt
  - requirements-gpu.txt
  - resultng
  - result-int8.png
  - save_modelpy
  - train.py
  - utils.py
  - External Libraries
  - Scratches and Consoles
  - library root
    - app.py
    - benchmarks.py
    - CODE_OF_CONDUCT.md
    - convert_tflite.py
    - convert_trt.py
    - detect.py
    - detectvideo.py
    - evaluate.py
    - LICENSE
    - README.md
    - requirements.txt
    - requirements-gpu.txt
    - resultng
    - result-int8.png
    - save_modelpy
    - train.py
    - utils.py
  - venv
    - libra
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
image = cv2.cvtColor(np.array(image), cv2.COLOR_BGR2RGB)
cv2.imwrite(FLAGS.output, image)

# convert image to byte type
with open(FLAGS.output, "rb") as image_file:
    encoded_string1 = base64.b64encode(image_file.read())

# convert byte to string
encoded_string = encoded_string1.decode("utf-8")
data["image"] = encoded_string
data["score"] = scores_arr
data["class"] = class_arr
data["success"] = True

return json.dumps(data, ensure_ascii=False, cls=my_utils.NumpyEncoder)

# route http posts to this method
@app.route('/api/detect', methods=['POST'])
def detect():
    data = {"success": False}
    if __name__ == "__main__":
        Use a production WSGI server instead.
        * Debug mode: off
        * Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
  - TODO
  - Terminal
  - Python Console
Event Log
2441 CRLF 4 spaces Python 3.6 (tensorflow-yolov4-tflite-master)
U: 0.0 KB/s D: 0.0 KB/s ENG 9:50 AM 06/08/2020

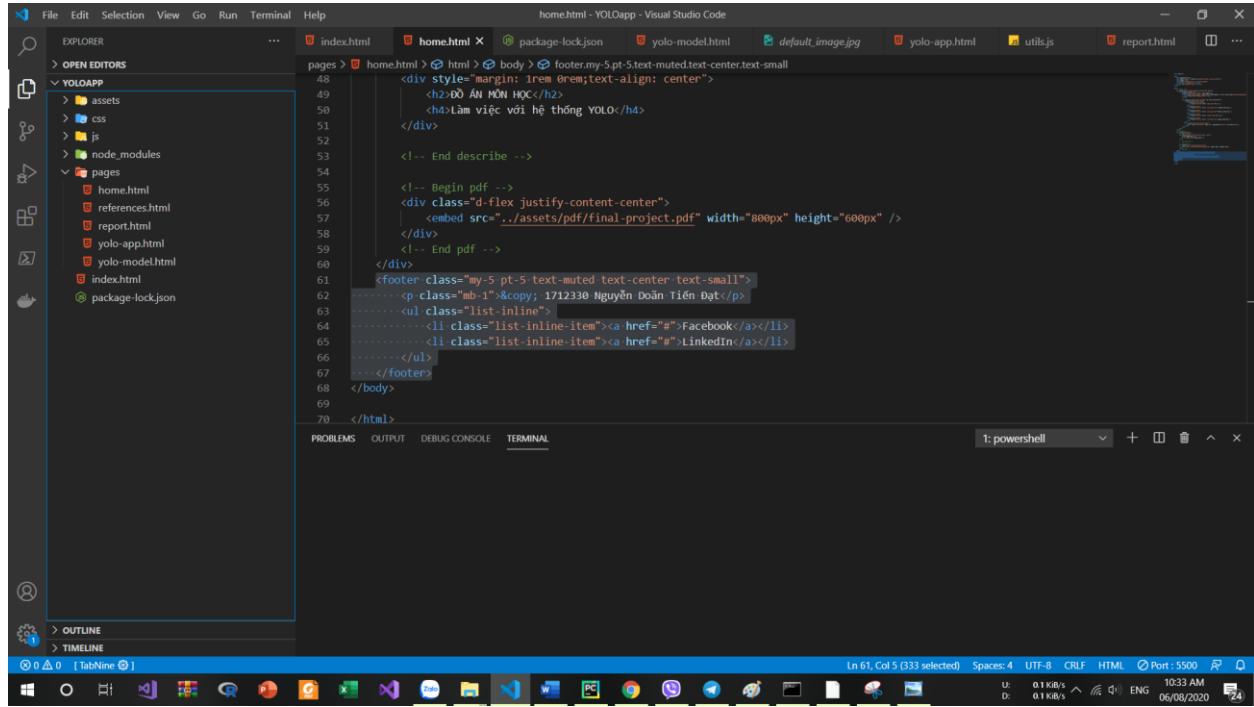
```

Để chạy server, chúng ta thực hiện câu lệnh:

```
python app.py
```

## VI. XÂY DỰNG GIAO DIỆN NGƯỜI DÙNG

Xây dựng giao diện đơn giản, với các chức năng quan trọng để tương tác với hệ thống YOLO của chúng ta: tải ảnh và hiển thị ảnh đã tải, gửi request cho server và nhận phản hồi từ server và hiển thị ảnh kết quả.



The screenshot shows the Visual Studio Code interface with the file 'home.html' open. The code is written in HTML and includes Bootstrap CSS classes like 'text-center' and 'list-inline'. It features a header with a title and subtitle, a PDF viewer component, and a footer with social media links. The Explorer sidebar on the left shows the project structure under 'YOLOAPP' with files like 'index.html', 'home.html', 'report.html', and 'yolo-model.html'. The bottom status bar shows system information including network speed and battery level.

```
<div style="margin: 1rem 0rem;text-align: center">
    <h2>ĐỒ ÁN MÔN HỌC</h2>
    <h4>Làm việc với hệ thống YOLO</h4>
</div>

<!-- End describe -->

<!-- Begin pdf -->
<div class="d-flex justify-content-center">
    <embed src="../assets/pdf/final-project.pdf" width="800px" height="600px" />
</div>
<!-- End pdf -->

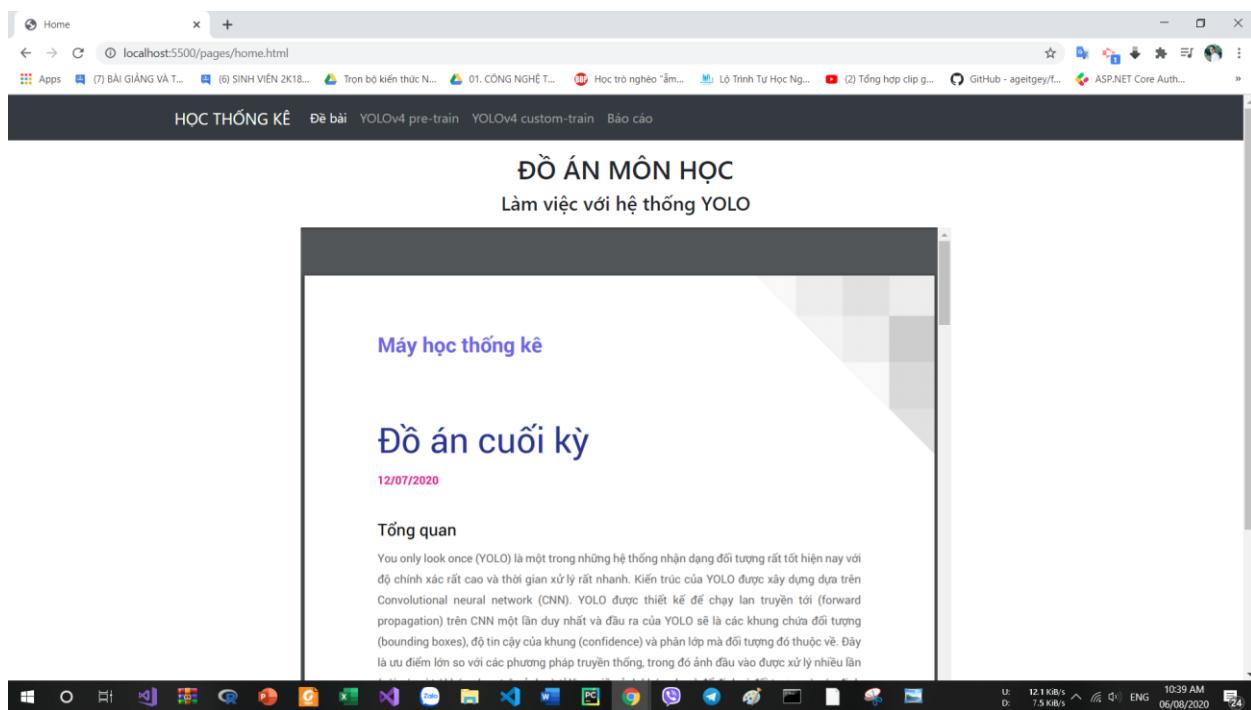
</div>
<footer class="my-5 pt-5 text-muted text-center text-small">
    <p class="mb-1">&copy; 1712330 Nguyễn Đoàn Tiến Đạt</p>
    <ul class="list-inline">
        <li class="list-inline-item"><a href="#">Facebook</a></li>
        <li class="list-inline-item"><a href="#">LinkedIn</a></li>
    </ul>
</footer>
</body>
</html>
```

Về phía client, chúng ta sử dụng thuận html, css, js để tạo giao diện tương tác phía người dùng kết hợp Bootstrap.

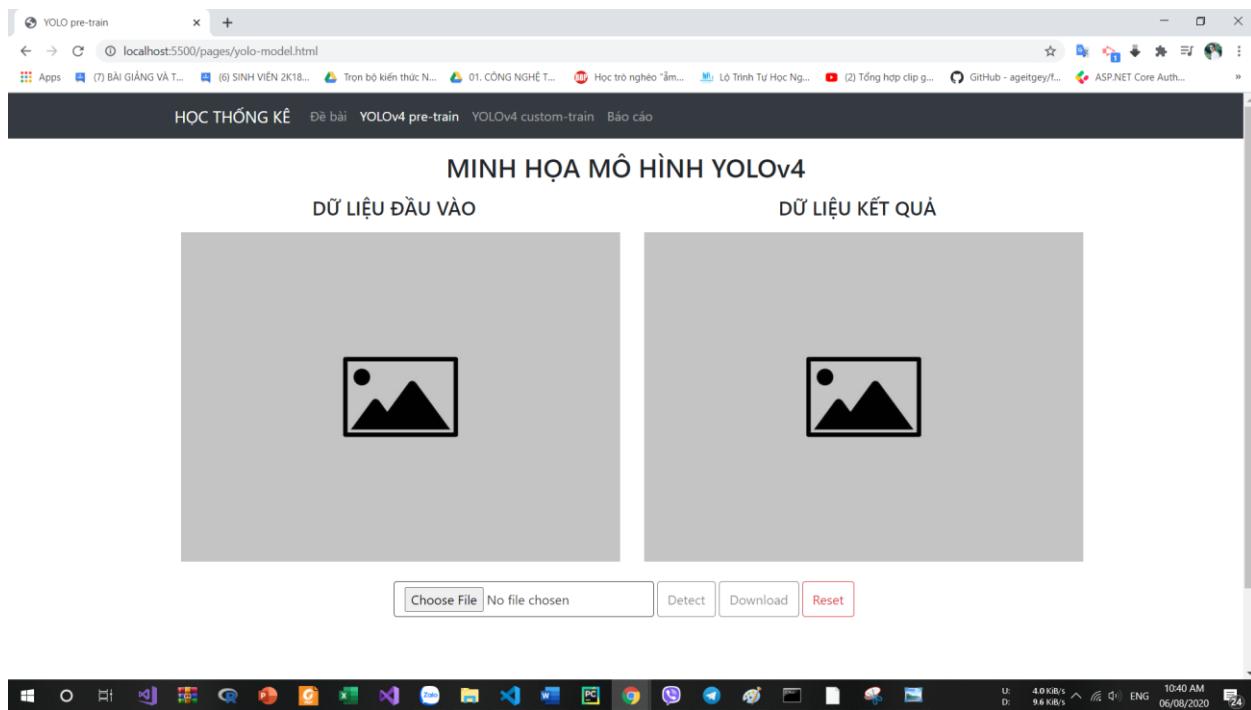
Giao diện của chúng ta sẽ có 4 phần: Đề bài, YOLO pre-train, YOLO custom-train, Báo cáo.

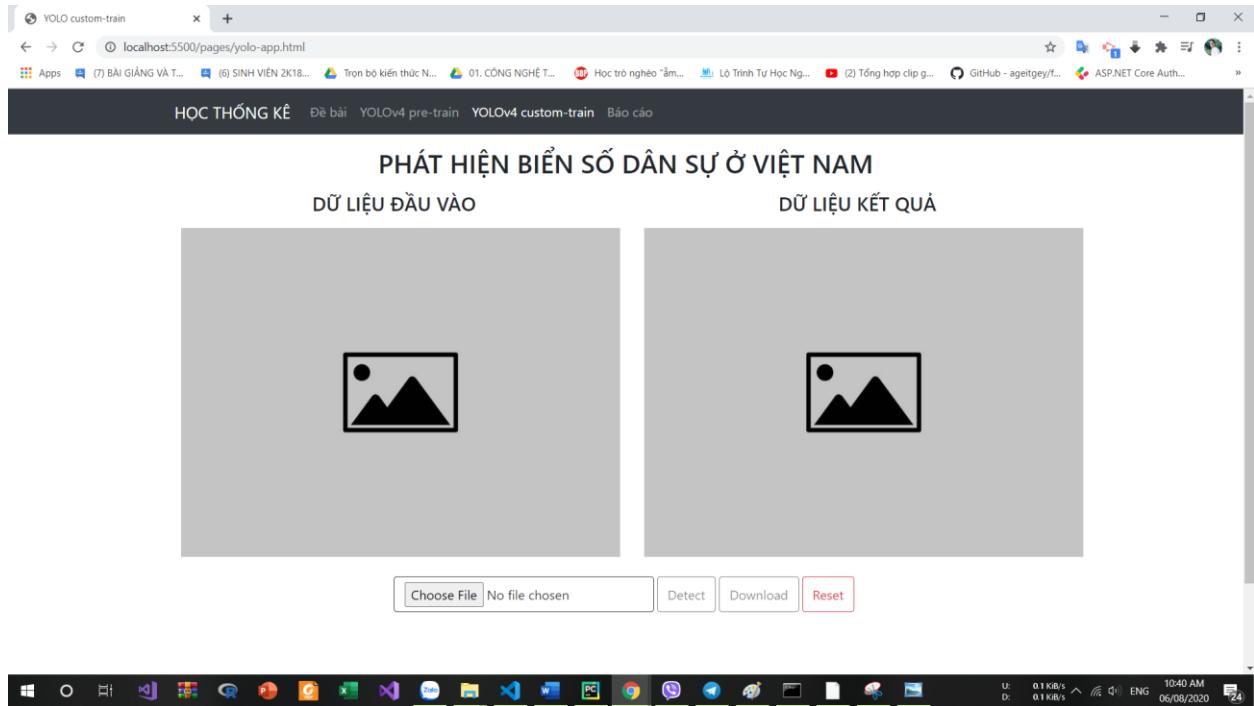
Cấu trúc mỗi trang: Header (chứa thanh điều hướng), Nội dung (chứa nội dung mỗi trang), Footer (chứa thông tin sinh viên).

Trang Đề bài và Báo cáo có giao diện tương đồng:

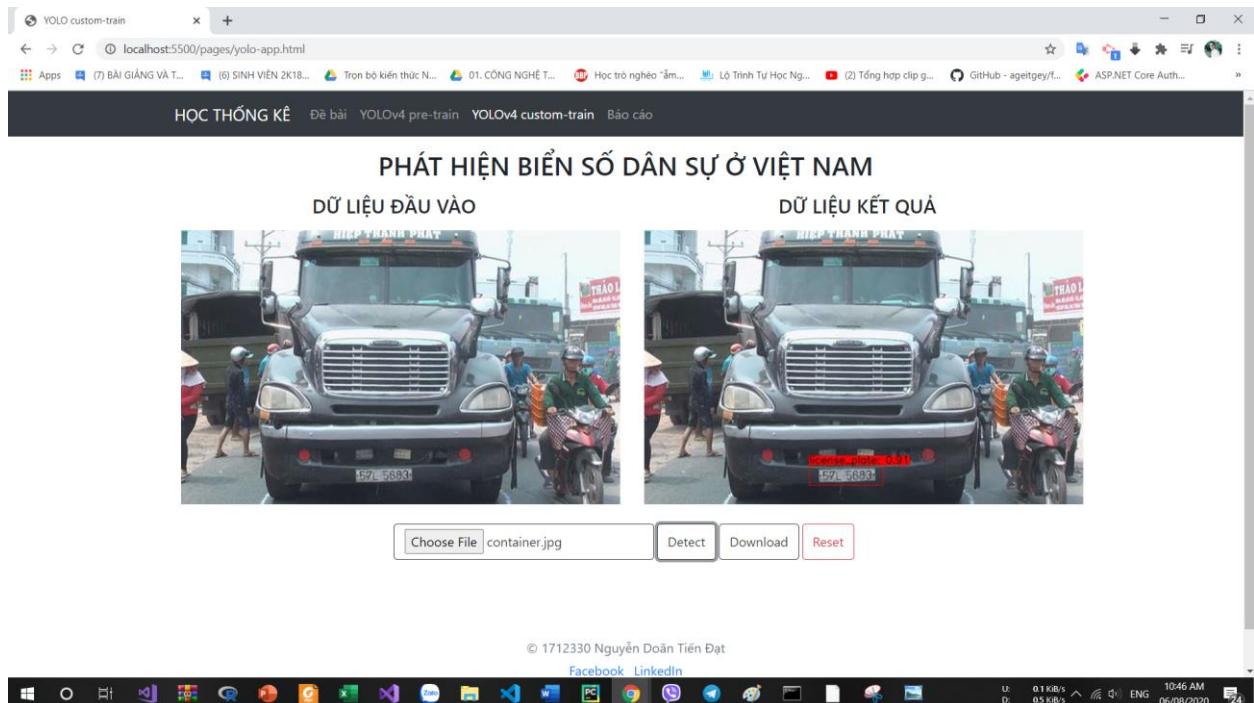


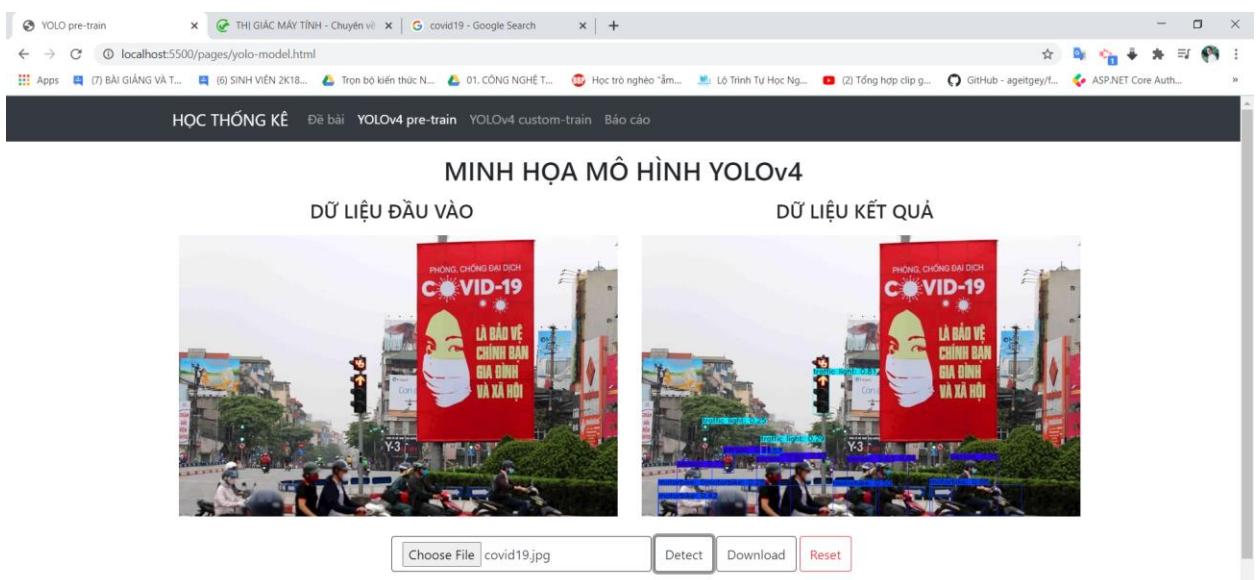
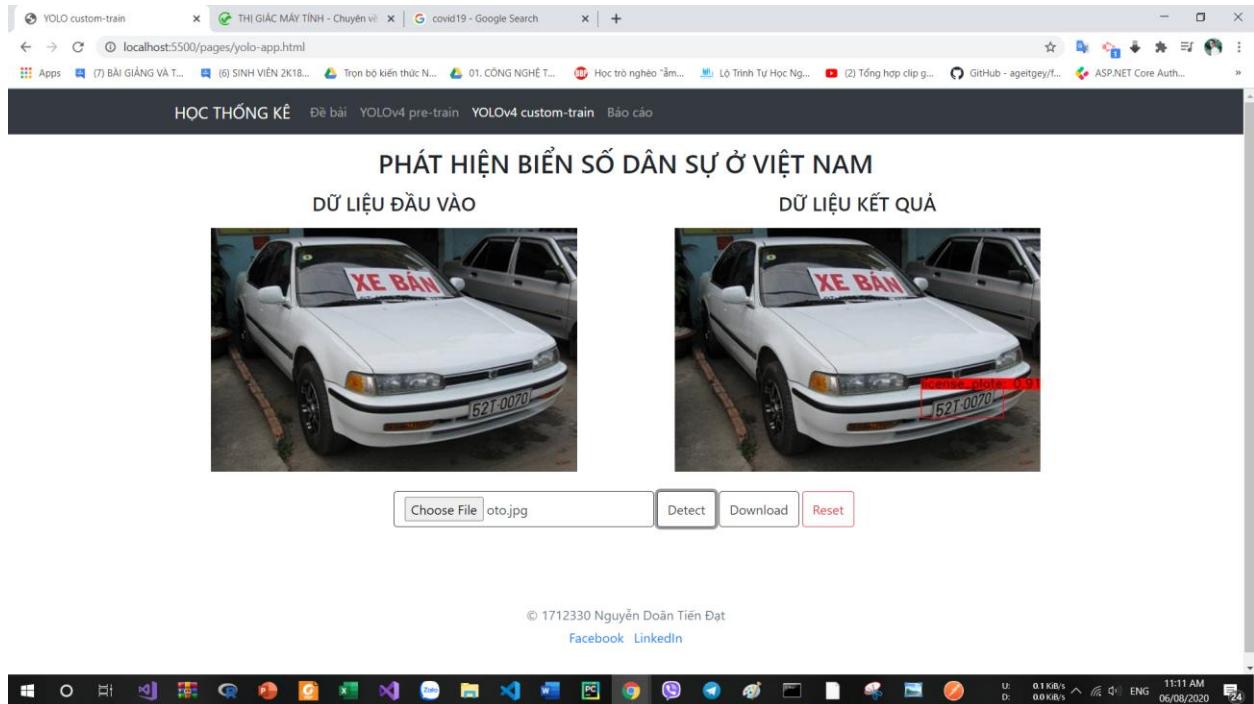
Trang YOLOv4 pretrain và YOLOv4 custom-train có giao diện tương đồng:





Demo sử dụng:





Một số lưu ý:

- Khi gửi request về server, chúng ta gửi nó dưới dạng FormData.
- Khi nhận ảnh chúng ta cần phải xử lý chuỗi base64 để hiển thị ảnh.
- Giao diện có thể tham khảo ở trên.

Nguồn source code: <https://github.com/tiendat3550/yolo-app.git>.

## VII. NGUỒN THAM KHẢO VÀ TÀI NGUYÊN

- [1] Joseph Redmon, Santosh Divvala, Ross Girshick, Ali Farhadi, “You Only Look Once: Unified, Real-Time Object Detection”, arXiv:1506.02640
- [2] Alexey Bochkovskiy, Chien-Yao Wang, Hong-Yuan Mark Liao, “YOLOv4: Optimal Speed and Accuracy of Object Detection”, arXiv:2004.10934.
- [3] Tzutalin, “LabelImg”, Git code: <https://github.com/tzutalin/labelImg>.
- [4] Phạm Đình Khánh, “Bài 29 - Xây dựng Flask API cho mô hình deep learning”, <https://phamdinhkhanh.github.io/2020/03/23/FlaskRestAPI.html>.
- [5] Phạm Đình Khánh, “Bài 25 - YOLO You Only Look Once” <https://phamdinhkhanh.github.io/2020/03/09/DarknetAlgorithm.html>.
- [6] Việt Hùng, “tensorflow-yolov4-tflite”, Git code: <https://github.com/hunglc007/tensorflow-yolov4-tflite>.
- [7] Võ Hùng Vũ, “Tổng hợp data”, <https://thigiacmaytinh.com/tai-nguyen-xu-ly-anh/tong-hop-data-xu-ly-anh/>.

--- HẾT ---