

ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN
KHOA CÔNG NGHỆ THÔNG TIN



MÁY HỌC THỐNG KÊ - TH2016/2

BÁO CÁO ĐỒ ÁN

YOU ONLY LOOK ONCE

❖ Thông tin sinh viên:

Họ và tên: Dương Đăng Khoa

MSSV: 1512251

GVHD: ThS. Ngô Minh Nhựt

Ngày 1 tháng 6 năm 2019

Mục Lục

I/ Cài đặt YOLO và sử dụng mô hình có sẵn	3
a/ Compile YOLO	3
b/ Xây dựng chương trình có giao diện cho phép đưa vào một bức ảnh và trả về kết quả nhận dạng đối tượng trên bức ảnh đó	5
II/ Tìm hiểu và huấn luyện để nhận dạng thêm một loại đối tượng mới	11
III/ Tài liệu tham khảo	18

I/ Cài đặt YOLO và sử dụng mô hình có sẵn

a/ Compile YOLO

- Bước 1: Mở Terminal và gõ lệnh `git clone https://github.com/pjreddie/darknet`
- Bước 2: Vào thư mục darknet vừa tải về bằng lệnh `cd darknet`
- Bước 3: Gõ lệnh `make` để biên dịch chương trình như hình bên dưới là thành công

```

khoadangduong@DANGKHOA:~$ cd Desktop/
khoadangduong@DANGKHOA:~/Desktop$ git clone https://github.com/pjreddie/darknet
Cloning into 'darknet'...
remote: Enumerating objects: 5901, done.
remote: Total 5901 (delta 0), reused 0 (delta 0), pack-reused 5901
Receiving objects: 100% (5901/5901), 6.16 MiB | 993.00 KiB/s, done.
Resolving deltas: 100% (3922/3922), done.
khoadangduong@DANGKHOA:~/Desktop$ cd darknet/
khoadangduong@DANGKHOA:~/Desktop/darknet$ make
mkdir -p obj
mkdir -p backup
mkdir -p results
gcc -Iinclude/ -Isrc/ -Wall -Wno-unused-result -Wno-unknown-pragmas -Wfatal-errors -fPIC -Ofast -c ./src/gemm.c -o obj/gemm.o
gcc -Iinclude/ -Isrc/ -Wall -Wno-unused-result -Wno-unknown-pragmas -Wfatal-errors -fPIC -Ofast -c ./src/utils.c -o obj/utils.o
gcc -Iinclude/ -Isrc/ -Wall -Wno-unused-result -Wno-unknown-pragmas -Wfatal-errors -fPIC -Ofast -c ./src/cuda.c -o obj/cuda.o
gcc -Iinclude/ -Isrc/ -Wall -Wno-unused-result -Wno-unknown-pragmas -Wfatal-errors -fPIC -Ofast -c ./src/deconvolutional_layer.c -o obj/deconvolutional_layer.o
gcc -Iinclude/ -Isrc/ -Wall -Wno-unused-result -Wno-unknown-pragmas -Wfatal-errors -fPIC -Ofast -c ./src/convolutional_layer.c -o obj/convolutional_layer.o
gcc -Iinclude/ -Isrc/ -Wall -Wno-unused-result -Wno-unknown-pragmas -Wfatal-errors -fPIC -Ofast -c ./src/list.c -o obj/list.o
gcc -Iinclude/ -Isrc/ -Wall -Wno-unused-result -Wno-unknown-pragmas -Wfatal-errors -fPIC -Ofast -c ./src/image.c -o obj/image.o
gcc -Iinclude/ -Isrc/ -Wall -Wno-unused-result -Wno-unknown-pragmas -Wfatal-errors -fPIC -Ofast -c ./src/activations.c -o obj/activations.o
gcc -Iinclude/ -Isrc/ -Wall -Wno-unused-result -Wno-unknown-pragmas -Wfatal-errors -fPIC -Ofast -c ./src/im2col.c -o obj/im2col.o
gcc -Iinclude/ -Isrc/ -Wall -Wno-unused-result -Wno-unknown-pragmas -Wfatal-errors -fPIC -Ofast -c ./src/col2im.c -o obj/col2im.o
gcc -Iinclude/ -Isrc/ -Wall -Wno-unused-result -Wno-unknown-pragmas -Wfatal-errors -fPIC -Ofast -c ./src/blas.c -o obj/blas.o
gcc -Iinclude/ -Isrc/ -Wall -Wno-unused-result -Wno-unknown-pragmas -Wfatal-errors -fPIC -Ofast -c ./src/crop_layer.c -o obj/crop_layer.o
gcc -Iinclude/ -Isrc/ -Wall -Wno-unused-result -Wno-unknown-pragmas -Wfatal-errors -fPIC -Ofast -c ./src/dropout_layer.c -o obj/dropout_layer.o
gcc -Iinclude/ -Isrc/ -Wall -Wno-unused-result -Wno-unknown-pragmas -Wfatal-errors -fPIC -Ofast -c ./src/maxpool_layer.c -o obj/maxpool_layer.o
gcc -Iinclude/ -Isrc/ -Wall -Wno-unused-result -Wno-unknown-pragmas -Wfatal-errors -fPIC -Ofast -c ./src/softmax_layer.c -o obj/softmax_layer.o
gcc -Iinclude/ -Isrc/ -Wall -Wno-unused-result -Wno-unknown-pragmas -Wfatal-errors -fPIC -Ofast -c ./src/data.c -o obj/data.o
gcc -Iinclude/ -Isrc/ -Wall -Wno-unused-result -Wno-unknown-pragmas -Wfatal-errors -fPIC -Ofast -c ./src/matrix.c -o obj/matrix.o
gcc -Iinclude/ -Isrc/ -Wall -Wno-unused-result -Wno-unknown-pragmas -Wfatal-errors -fPIC -Ofast -c ./src/network.c -o obj/network.o
gcc -Iinclude/ -Isrc/ -Wall -Wno-unused-result -Wno-unknown-pragmas -Wfatal-errors -fPIC -Ofast -c ./src/connected_layer.c -o obj/connected_layer.o
gcc -Iinclude/ -Isrc/ -Wall -Wno-unused-result -Wno-unknown-pragmas -Wfatal-errors -fPIC -Ofast -c ./src/cost_layer.c -o obj/cost_layer.o
gcc -Iinclude/ -Isrc/ -Wall -Wno-unused-result -Wno-unknown-pragmas -Wfatal-errors -fPIC -Ofast -c ./src/parser.c -o obj/parser.o
gcc -Iinclude/ -Isrc/ -Wall -Wno-unused-result -Wno-unknown-pragmas -Wfatal-errors -fPIC -Ofast -c ./src/region_layer.c -o obj/region_layer.o
gcc -Iinclude/ -Isrc/ -Wall -Wno-unused-result -Wno-unknown-pragmas -Wfatal-errors -fPIC -Ofast -c ./src/region_list.c -o obj/region_list.o
gcc -Iinclude/ -Isrc/ -Wall -Wno-unused-result -Wno-unknown-pragmas -Wfatal-errors -fPIC -Ofast -c ./src/region_list.c -o obj/region_list.o

```

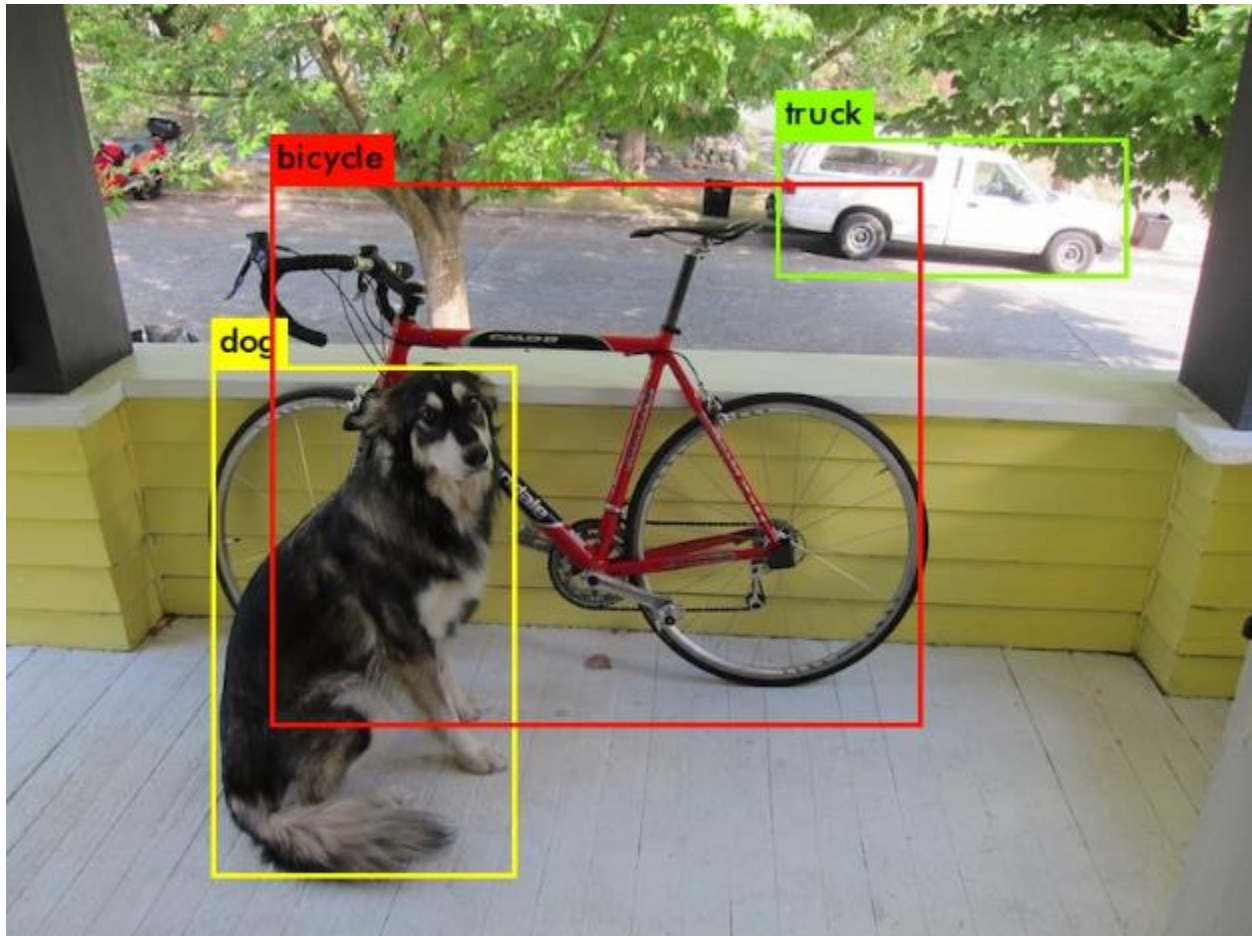
- Bước 4: Tiếp tục nhập dòng sau vào Terminal để tải yolov3.weights
`wget https://pjreddie.com/media/files/yolov3.weights`
- Bước 5: Thực hiện dòng lệnh sau để test thử thuật toán YOLO
`./darknet detect cfg/yolov3.cfg yolov3.weights data/dog.jpg`
- Kết quả trả về như hình bên dưới

```
khoadangduong@DANGKHOA: ~/Desktop/darknet
./darknet detector test cfg/coco.data cfg/yolov3.cfg yolov3.weights data/dog.jpg
layer   filters  size      input              output              BFLOPs
0 conv   32  3 x 3 / 1  608 x 608 x 3  ->  608 x 608 x 32  0.639
1 conv   64  3 x 3 / 2  608 x 608 x 32  ->  304 x 304 x 64  3.407
2 conv   32  1 x 1 / 1  304 x 304 x 64  ->  304 x 304 x 32  0.379
3 conv   64  3 x 3 / 1  304 x 304 x 32  ->  304 x 304 x 64  3.407
4 res    1                304 x 304 x 64  ->  304 x 304 x 64
5 conv   128 3 x 3 / 2  304 x 304 x 64  ->  152 x 152 x 128 3.407
6 conv   64  1 x 1 / 1  152 x 152 x 128 ->  152 x 152 x 64  0.379
7 conv   128 3 x 3 / 1  152 x 152 x 64  ->  152 x 152 x 128 3.407
8 res    5                152 x 152 x 128 ->  152 x 152 x 128
9 conv   64  1 x 1 / 1  152 x 152 x 128 ->  152 x 152 x 64  0.379
10 conv  128 3 x 3 / 1  152 x 152 x 64  ->  152 x 152 x 128 3.407
11 res   8                152 x 152 x 128 ->  152 x 152 x 128
12 conv  256 3 x 3 / 2  152 x 152 x 128 ->  76 x 76 x 256  3.407
13 conv  128 1 x 1 / 1  76 x 76 x 256   ->  76 x 76 x 128  0.379
14 conv  256 3 x 3 / 1  76 x 76 x 128   ->  76 x 76 x 256  3.407
15 res  12                76 x 76 x 256   ->  76 x 76 x 256
16 conv  128 1 x 1 / 1  76 x 76 x 256   ->  76 x 76 x 128  0.379
17 conv  256 3 x 3 / 1  76 x 76 x 128   ->  76 x 76 x 256  3.407
18 res  15                76 x 76 x 256   ->  76 x 76 x 256
19 conv  128 1 x 1 / 1  76 x 76 x 256   ->  76 x 76 x 128  0.379
20 conv  256 3 x 3 / 1  76 x 76 x 128   ->  76 x 76 x 256  3.407
21 res  18                76 x 76 x 256   ->  76 x 76 x 256
22 conv  128 1 x 1 / 1  76 x 76 x 256   ->  76 x 76 x 128  0.379
23 conv  256 3 x 3 / 1  76 x 76 x 128   ->  76 x 76 x 256  3.407
24 res  21                76 x 76 x 256   ->  76 x 76 x 256
25 conv  128 1 x 1 / 1  76 x 76 x 256   ->  76 x 76 x 128  0.379
26 conv  256 3 x 3 / 1  76 x 76 x 128   ->  76 x 76 x 256  3.407
27 res  24                76 x 76 x 256   ->  76 x 76 x 256
28 conv  128 1 x 1 / 1  76 x 76 x 256   ->  76 x 76 x 128  0.379
29 conv  256 3 x 3 / 1  76 x 76 x 128   ->  76 x 76 x 256  3.407
30 res  27                76 x 76 x 256   ->  76 x 76 x 256
31 conv  128 1 x 1 / 1  76 x 76 x 256   ->  76 x 76 x 128  0.379
32 conv  256 3 x 3 / 1  76 x 76 x 128   ->  76 x 76 x 256  3.407
33 conv  256 3 x 3 / 1  76 x 76 x 128   ->  76 x 76 x 256  3.407

78 conv  1024 3 x 3 / 1  19 x 19 x 512  ->  19 x 19 x 1024  3.407
79 conv   512 1 x 1 / 1  19 x 19 x 1024 ->  19 x 19 x 512  0.379
80 conv  1024 3 x 3 / 1  19 x 19 x 512  ->  19 x 19 x 1024  3.407
81 conv   255 1 x 1 / 1  19 x 19 x 1024 ->  19 x 19 x 255  0.189
82 yolo
83 route 79
84 conv   256 1 x 1 / 1  19 x 19 x 512  ->  19 x 19 x 256  0.095
85 upsample 2x      19 x 19 x 256  ->  38 x 38 x 256
86 route 85 61
87 conv   256 1 x 1 / 1  38 x 38 x 768  ->  38 x 38 x 256  0.568
88 conv   512 3 x 3 / 1  38 x 38 x 256  ->  38 x 38 x 512  3.407
89 conv   256 1 x 1 / 1  38 x 38 x 512  ->  38 x 38 x 256  0.379
90 conv   512 3 x 3 / 1  38 x 38 x 256  ->  38 x 38 x 512  3.407
91 conv   256 1 x 1 / 1  38 x 38 x 512  ->  38 x 38 x 256  0.379
92 conv   512 3 x 3 / 1  38 x 38 x 256  ->  38 x 38 x 512  3.407
93 conv   255 1 x 1 / 1  38 x 38 x 512  ->  38 x 38 x 255  0.377
94 yolo
95 route 91
96 conv   128 1 x 1 / 1  38 x 38 x 256  ->  38 x 38 x 128  0.095
97 upsample 2x      38 x 38 x 128  ->  76 x 76 x 128
98 route 97 36
99 conv   128 1 x 1 / 1  76 x 76 x 384  ->  76 x 76 x 128  0.568
100 conv  256 3 x 3 / 1  76 x 76 x 128  ->  76 x 76 x 256  3.407
101 conv  128 1 x 1 / 1  76 x 76 x 256  ->  76 x 76 x 128  0.379
102 conv  256 3 x 3 / 1  76 x 76 x 128  ->  76 x 76 x 256  3.407
103 conv  128 1 x 1 / 1  76 x 76 x 256  ->  76 x 76 x 128  0.379
104 conv  256 3 x 3 / 1  76 x 76 x 128  ->  76 x 76 x 256  3.407
105 conv  255 1 x 1 / 1  76 x 76 x 256  ->  76 x 76 x 255  0.754
106 yolo

Loading weights from yolov3.weights...Done!
data/dog.jpg: Predicted in 27.948396 seconds.
dog: 100%
truck: 92%
bicycle: 99%
khoadangduong@DANGKHOA:~/Desktop/darknet$
```

- Sau khi thực hiện xong sẽ xuất hiện kết quả như hình trên với tỉ lệ dự đoán rất chính xác với dog: 100%, truck: 92%, bicycle: 99%



b/ Xây dựng chương trình có giao diện cho phép đưa vào một bức ảnh và trả về kết quả nhận dạng đối tượng trên bức ảnh đó

- Vào thư mục darknet/python và tạo file ui.py để tạo giao diện cho chương trình đồng thời viết các hàm để cho phép chương trình đưa vào một bức ảnh và trả về kết quả nhận dạng đối tượng trên bức ảnh đó:
- Dưới đây là hình ảnh của giao diện:
 - Sử dụng python 2.7 để chạy file ui.py. Mở Terminal trong thư mục darknet/python và gõ lệnh `python ui.py` để chạy ta được một giao diện như hình bên dưới

Activities Tk ▾ Thu Jun 6, 22:33 en ▾ 100% ▾
khoadangduong@DANGKHOA: ~/Desktop/Bai 1/python

File Edit View Search Terminal Help

```
khoadangduong@DANGKHOA:~/Desktop/Bai 1/python$ python ut.py
```

layer	filters	size	input	output
0 conv	32	3 x 3 / 1	416 x 416 x 3	208 x 208 x 32 0.299 BFLOPs
1 conv	64	3 x 3 / 2	416 x 416 x 32	208 x 208 x 64 1.595 BFLOPs
2 conv	32	1 x 1 / 1	208 x 208 x 64	208 x 208 x 32 0.177 BFLOPs
3 conv	64	3 x 3 / 1	208 x 208 x 32	208 x 208 x 64 1.595 BFLOPs
4 res	1		208 x 208 x 64	208 x 208 x 64
5 conv	128	3 x 3 / 2	208 x 208 x 64	104 x 104 x 128 1.595 BFLOPs
6 conv	64	1 x 1 / 1	104 x 104 x 128	104 x 104 x 64 0.177 BFLOPs
7 conv	128	3 x 3 / 1	104 x 104 x 64	104 x 104 x 128 1.595 BFLOPs
8 res	5		104 x 104 x 128	104 x 104 x 128
9 conv	64	1 x 1 / 1	104 x 104 x 128	104 x 104 x 64 0.177 BFLOPs
10 conv	128	3 x 3 / 1	104 x 104 x 64	104 x 104 x 128 1.595 BFLOPs
11 res	8		104 x 104 x 128	104 x 104 x 128
12 conv	256	3 x 3 / 2	104 x 104 x 128	52 x 52 x 256 1.595 BFLOPs
13 conv	128	1 x 1 / 1	52 x 52 x 256	52 x 52 x 128 0.177 BFLOPs
14 conv	256	3 x 3 / 1	52 x 52 x 128	52 x 52 x 256 1.595 BFLOPs
15 res	12		52 x 52 x 256	52 x 52 x 256
16 conv	128	1 x 1 / 1	52 x 52 x 256	52 x 52 x 128 0.177 BFLOPs
17 conv	256	3 x 3 / 1	52 x 52 x 128	52 x 52 x 256 1.595 BFLOPs
18 res	15		52 x 52 x 256	52 x 52 x 256
19 conv	128	1 x 1 / 1	52 x 52 x 256	52 x 52 x 128 0.177 BFLOPs
20 conv	256	3 x 3 / 1	52 x 52 x 128	52 x 52 x 256 1.595 BFLOPs
21 res	18		52 x 52 x 256	52 x 52 x 256
22 conv	128	1 x 1 / 1	52 x 52 x 256	52 x 52 x 128 0.177 BFLOPs
23 conv	256	3 x 3 / 1	52 x 52 x 128	52 x 52 x 256 1.595 BFLOPs
24 res	21		52 x 52 x 256	52 x 52 x 256
25 conv	128	1 x 1 / 1	52 x 52 x 256	52 x 52 x 128 0.177 BFLOPs
26 conv	256	3 x 3 / 1	52 x 52 x 128	52 x 52 x 256 1.595 BFLOPs
27 res	24		52 x 52 x 256	52 x 52 x 256
28 conv	128	1 x 1 / 1	52 x 52 x 256	52 x 52 x 128 0.177 BFLOPs
29 conv	256	3 x 3 / 1	52 x 52 x 128	52 x 52 x 256 1.595 BFLOPs
30 res	27		52 x 52 x 256	52 x 52 x 256
31 conv	128	1 x 1 / 1	52 x 52 x 256	52 x 52 x 128 0.177 BFLOPs
32 conv	256	3 x 3 / 1	52 x 52 x 128	52 x 52 x 256 1.595 BFLOPs
33 conv	256	3 x 3 / 1	52 x 52 x 256	52 x 52 x 256

You Only Look Once - YOLO - Duong Dang Khoa - 151251

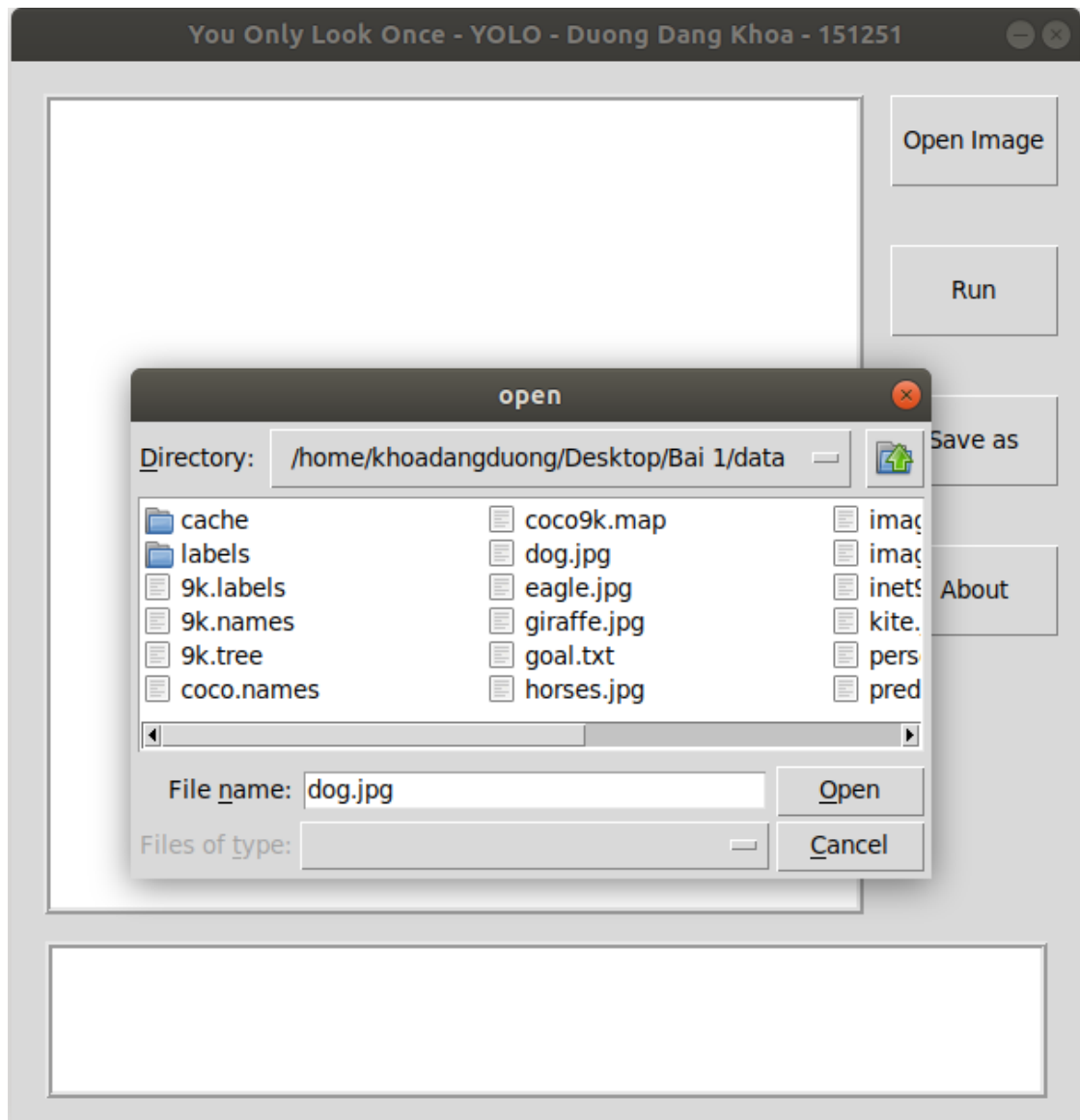
Open Image

Run

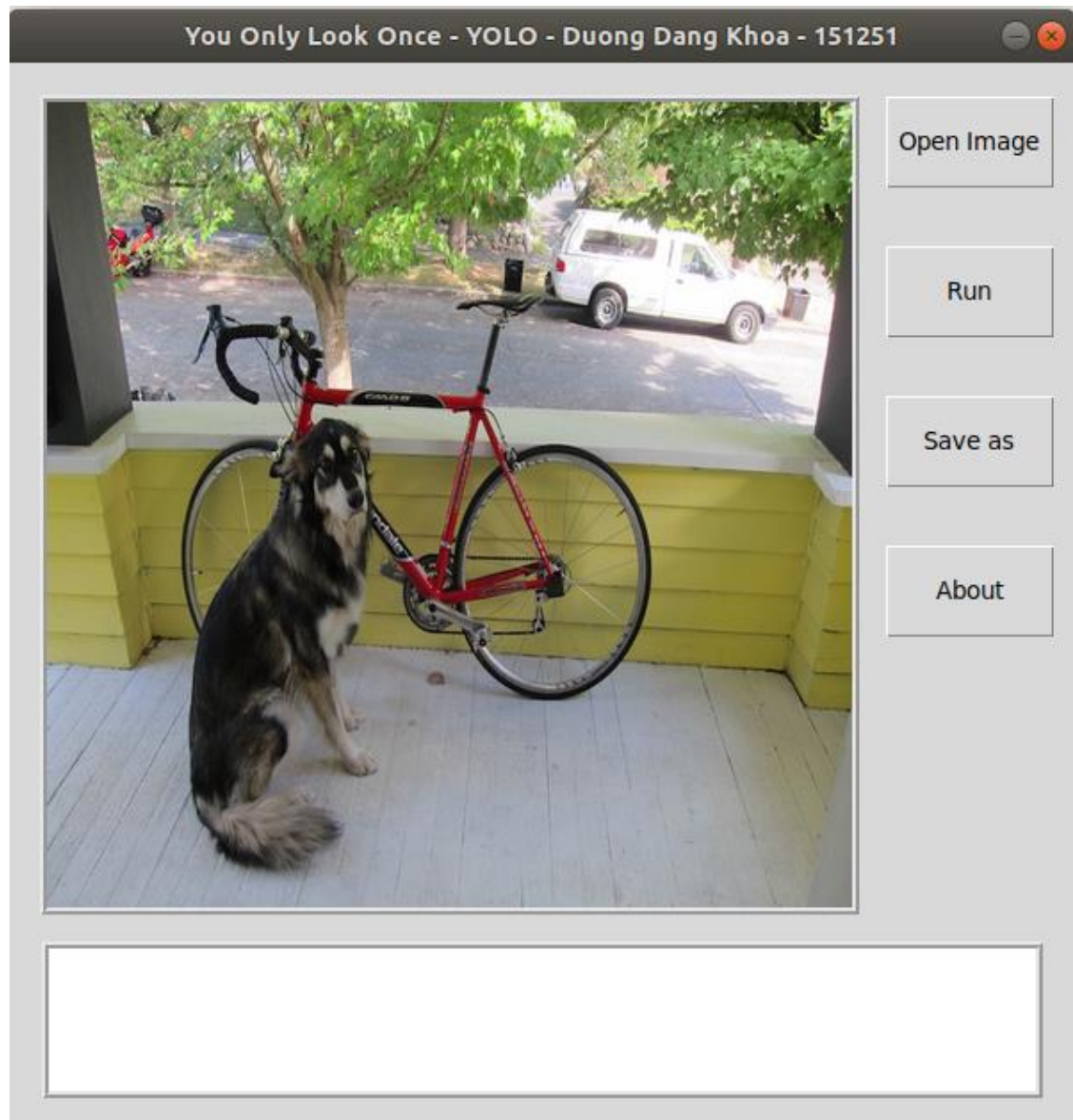
Save as

About

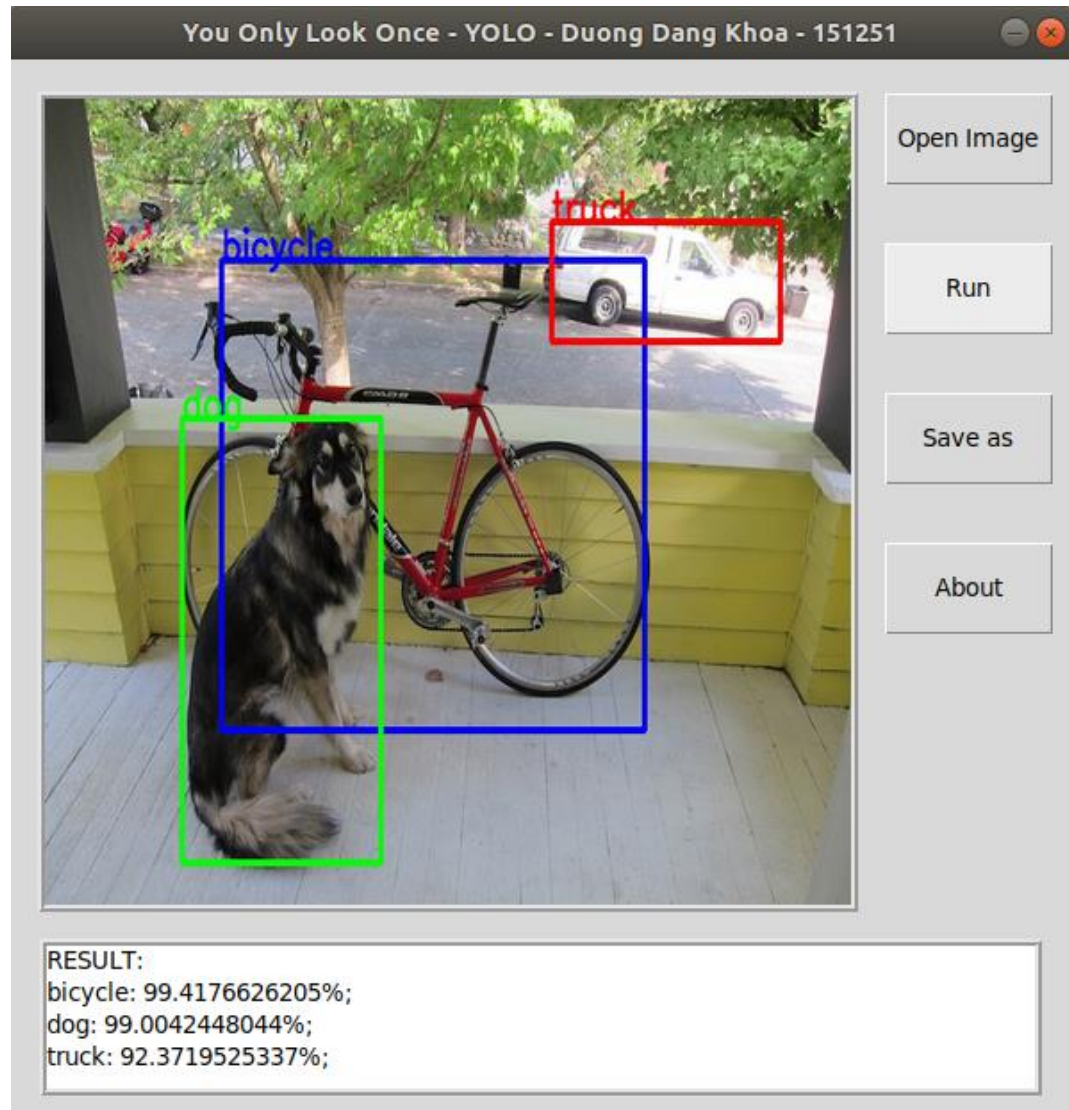
- Mô tả giao diện:
 - Có 1 frame lớn dùng để chứa ảnh đã chọn và 1 listbox dùng để hiển thị kết quả
 - Có 4 chức năng chính với 4 buttons: Open Image, Run, Save as, About
 - ✓ Nút Open Image: Dùng để mở chọn file ảnh để load vào hệ thống



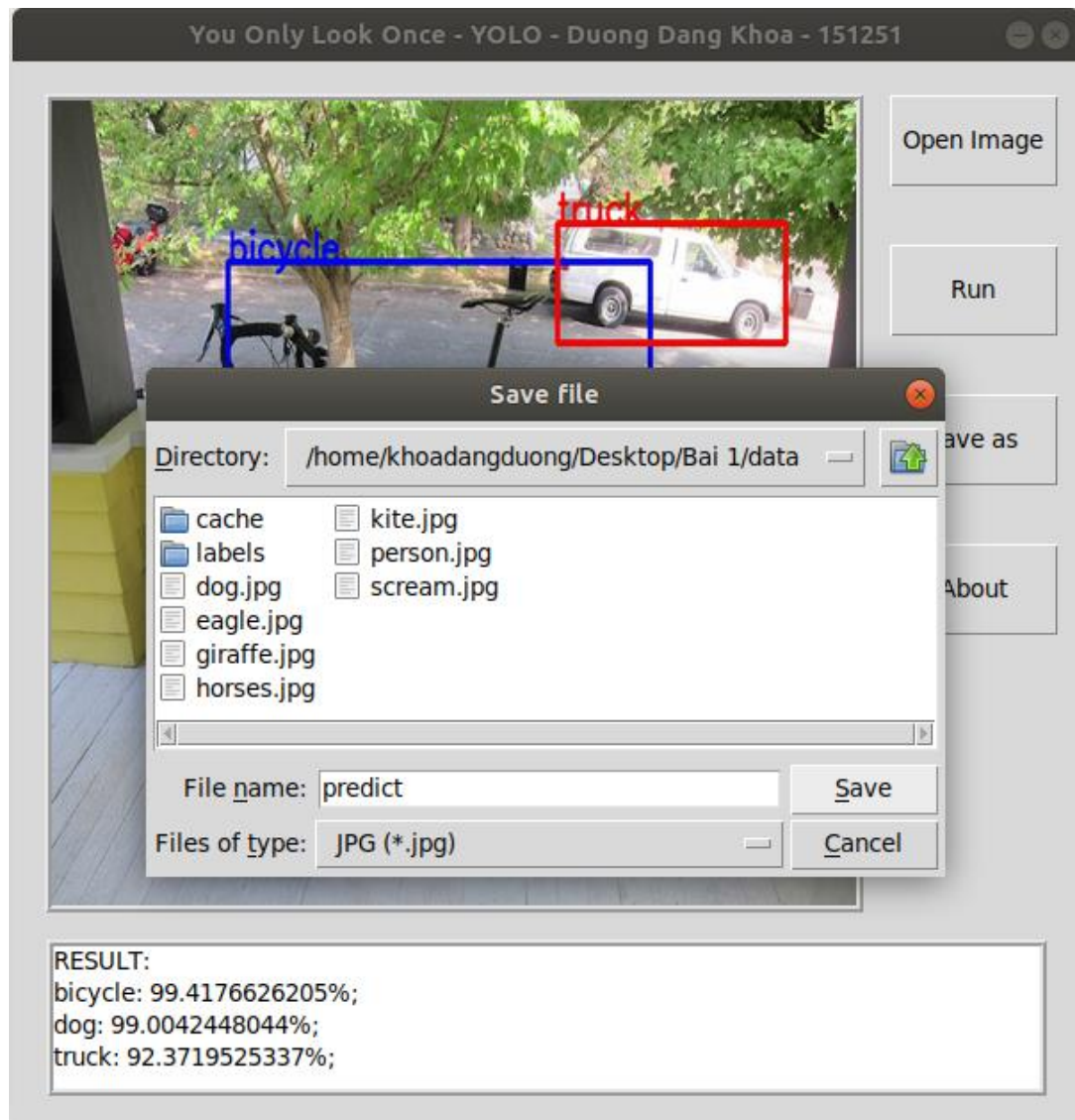
Sau khi chọn hình ảnh và bấm Open thì hình ảnh sẽ được load vào form giao diện



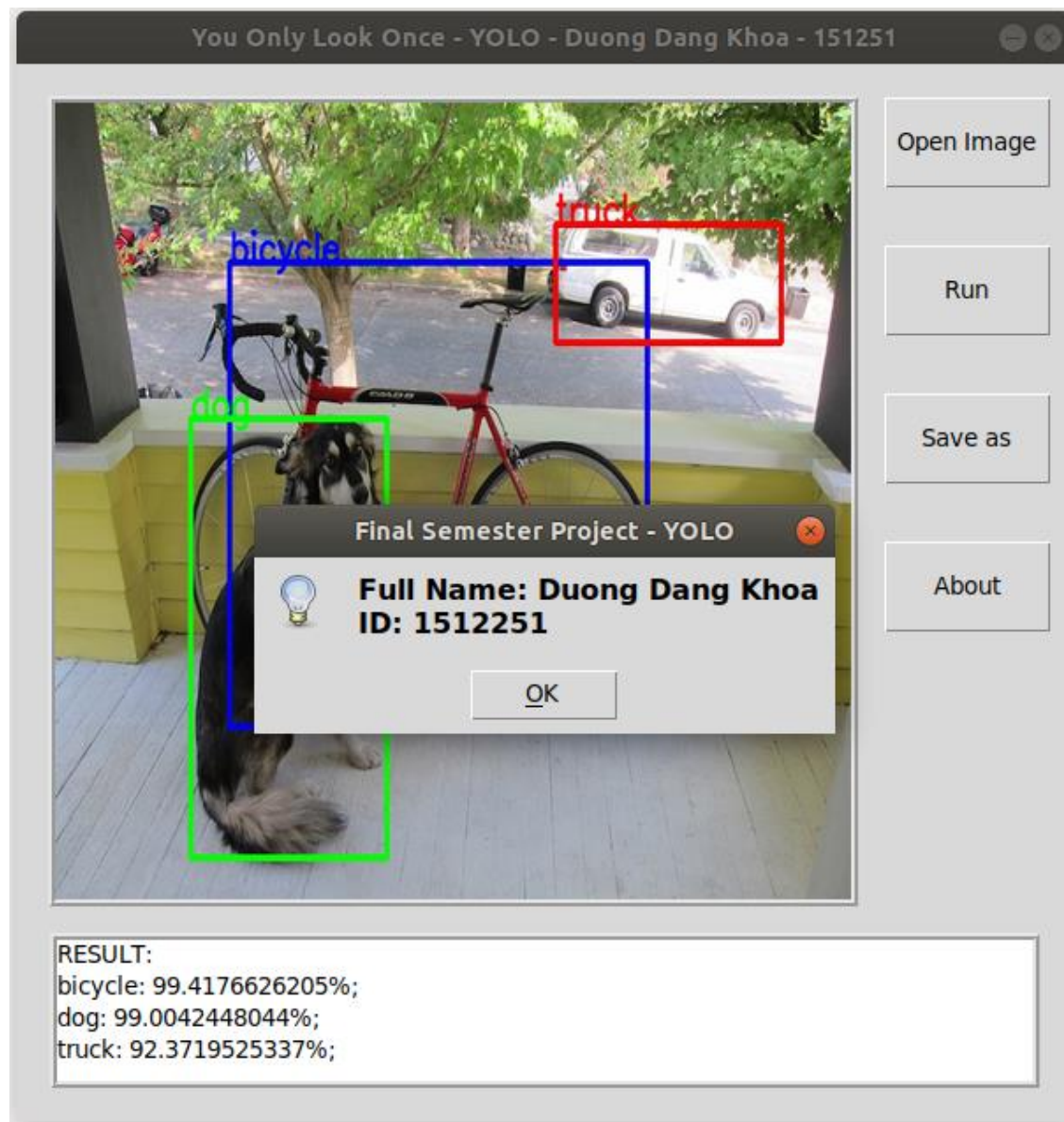
- ✓ Nút Run: Dùng để detect hình ảnh đó. Đợi khoảng một chút các bounding box sẽ xuất hiện trên hình



- ✓ Nút Save as: Dùng khi có nhu cầu lưu trữ lại hình ảnh vừa detect được



- ✓ Nút About: Dùng để xem thông tin nhà phát triển.



II/ Tìm hiểu và huấn luyện để nhận dạng thêm một loại đối tượng mới

- **Yêu cầu:** Tìm hiểu các huấn luyện YOLO và thêm dữ liệu về một đối tượng mới vào để huấn luyện ra một mô hình mới có khả năng nhận dạng đối tượng mới này.
- **Chọn đối tượng mới:** Phân loại 5 ngón tay và 0 ngón tay
- **Tập dữ liệu:** 636 ảnh bao gồm ảnh bàn tay 5 ngón và nắm tay tượng trưng cho 0 ngón
- **Ứng dụng:**
 - Có thể phát triển xây dựng app chơi game oẳn tù xì khi huấn luyện thêm 1 lớp 2 ngón tay tượng trưng cho kéo, 0 ngón tay tượng trưng cho búa, 5 ngón tay tượng trưng cho bao

- Phân tích hành vi phát hiện các hành động nắm, tát, đá ... ứng dụng trong việc chấm điểm khi thi đấu võ đài
- Làm phần mềm trên các thiết bị thông minh gắn kèm trên người cảnh sát để phân tích các hành vi cảnh báo hành động nguy hiểm của đối tượng khi đang làm nhiệm vụ

- **Thực hiện:**

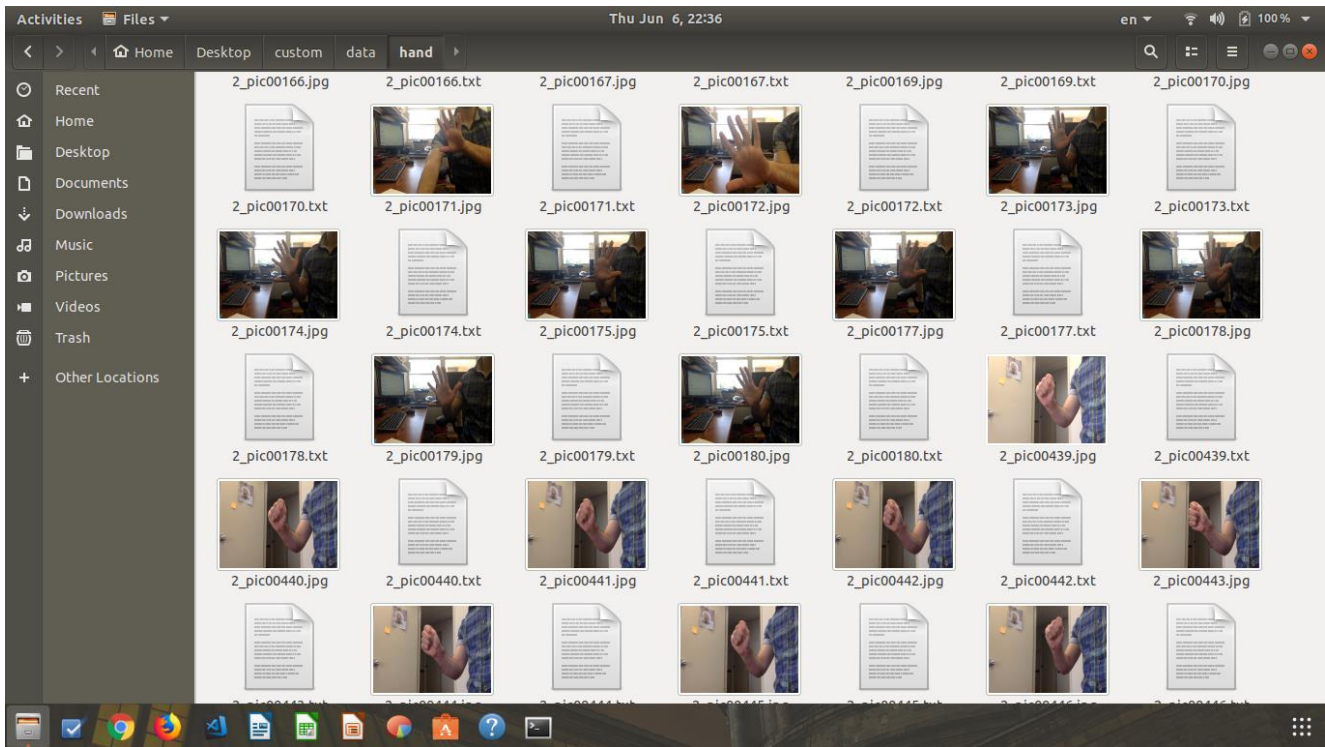
- Bước 1: Mở terminal và thực hiện dòng lệnh sau:
`git clone https://github.com/ManivannanMurugavel/YOLO-Annotation-Tool.git`
- Bước 2: Sửa classes.txt thêm vào 2 đối tượng là 5_fingers và 0_finger
- Bước 3: Gõ lệnh `python main.py` trên Terminal trong thư mục YOLO-Annotation-Tool.git và bắt đầu đánh nhãn

Ta vẽ khung bao quanh hình cái bàn như hình bên dưới, sau khi vẽ chương trình sẽ tự động lưu lại tọa độ của khung bao vào một file text có tên giống với tên của hình ảnh và nội dung có dạng

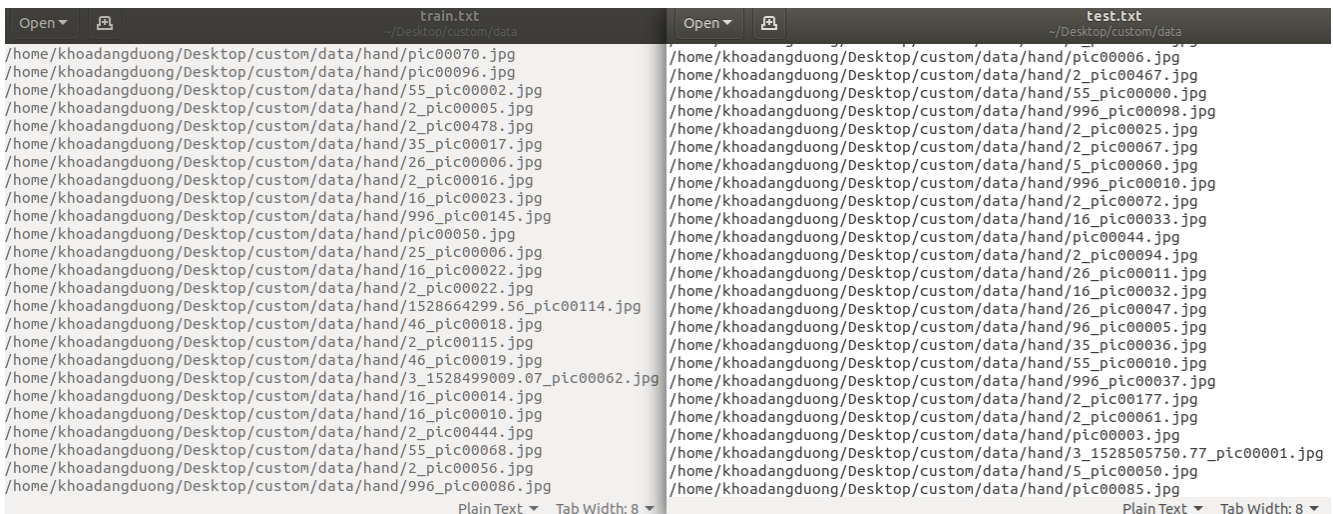
*[category number] [bounding box left X] [bounding box top Y] [bounding box right X]
[bounding box bottom Y]*



Ta tiếp tục thực hiện hành động tương tự cho đến hết tất cả hình ảnh cần train



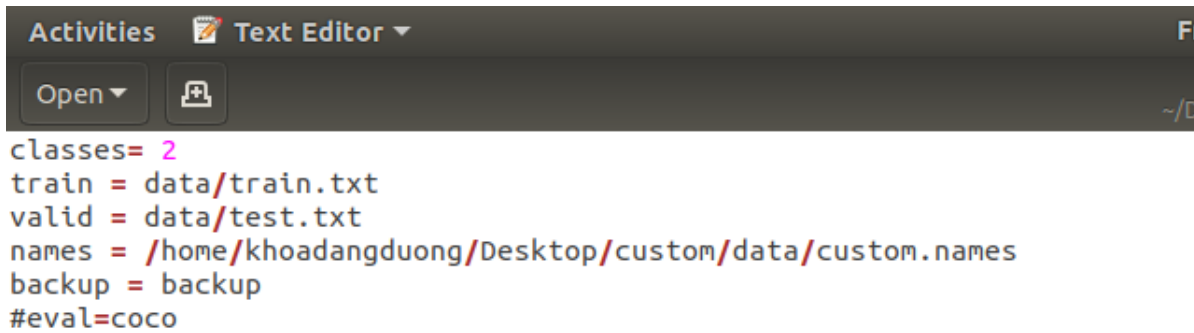
- Bước 4: Sau khi thực thi file process.py ta sẽ có được 2 file train.txt và test.txt nằm ở thư mục YOLO-Annotation-Tool và nội dung như bên dưới. Sau đó ta sẽ copy 2 file đó vào thư mục darknet.



- Bước 5: Tiếp theo ta sẽ tạo file custom.data và custom.names trong thư mục darknet/cfg với nội dung như sau:

File custom.names sẽ có 2 class 5_fingers và 0_finger

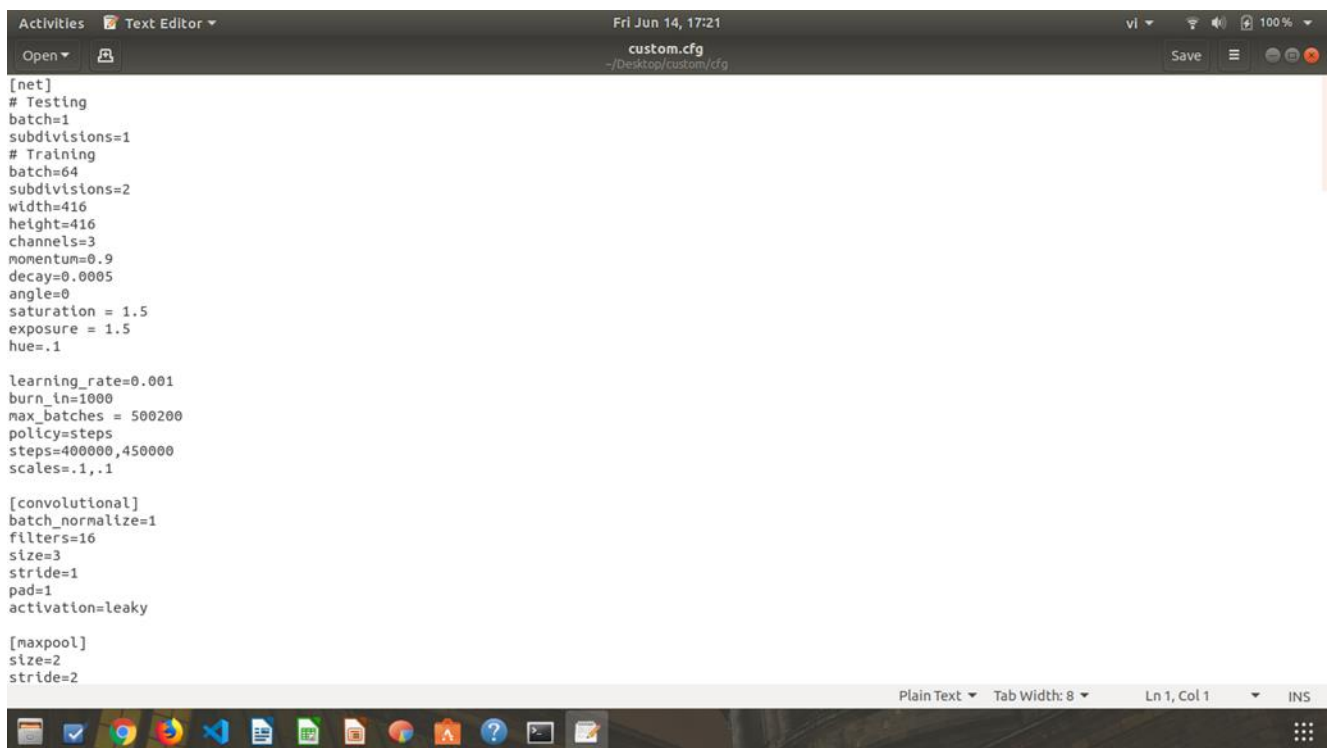
File custom.data:



```

classes= 2
train = data/train.txt
valid = data/test.txt
names = /home/khoadangduong/Desktop/custom/data/custom.names
backup = backup
#eval=coco
    
```

- Bước 6: Ta sẽ sử dụng file yolov3.cfg để train và ta sẽ thay đổi một vài thông số trong file này như sau:



```

[net]
# Testing
batch=1
subdivisions=1
# Training
batch=64
subdivisions=2
width=416
height=416
channels=3
momentum=0.9
decay=0.0005
angle=0
saturation = 1.5
exposure = 1.5
hue=.1

learning_rate=0.001
burn_in=1000
max_batches = 500200
policy=steps
steps=400000,450000
scales=.1,.1

[convolutional]
batch_normalize=1
filters=16
size=3
stride=1
pad=1
activation=leaky

[maxpool]
size=2
stride=2
    
```

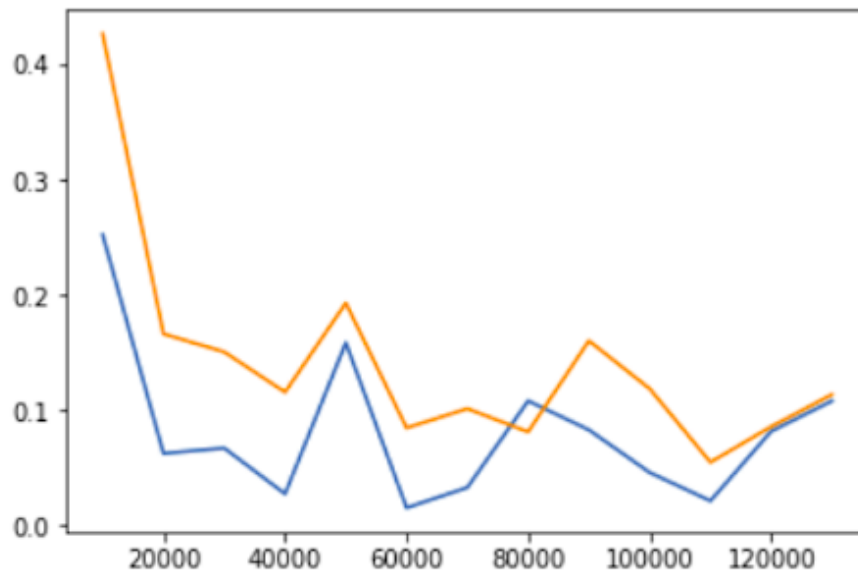
- Bước 7: Tiếp theo ta sẽ tải một file weights khởi đầu có sẵn để bắt đầu train cho dữ liệu cho đến khi hội tụ. Ta sử dụng file darknet53.conv.74. Có thể tải file này từ dòng lệnh bên dưới:
`wget https://pjreddie.com/media/files/darknet53.conv.74`
- Bước 8: Ta mở terminal tới thư mục darknet và thực thi dòng lệnh bên dưới để bắt đầu train
`./darknet detector train cfg/custom.data cfg/custom.cfg darknet53.conv.74`
Quá trình train sẽ khá tốn thời gian. Sau mỗi 100 vòng lặp thì chương trình sẽ tự động lưu file yolo3_xxx.weights (với xxx là số vòng lặp) vào thư mục backup trong darknet
- Bước 9: Ta mở file detector.c và thay đổi dòng bên dưới
`if(i%10000==0 || (i < 1000 && i%100 == 0))`

thành dòng

```
if(i%10000==0 || (i < 10000 && i%100 == 0))
```

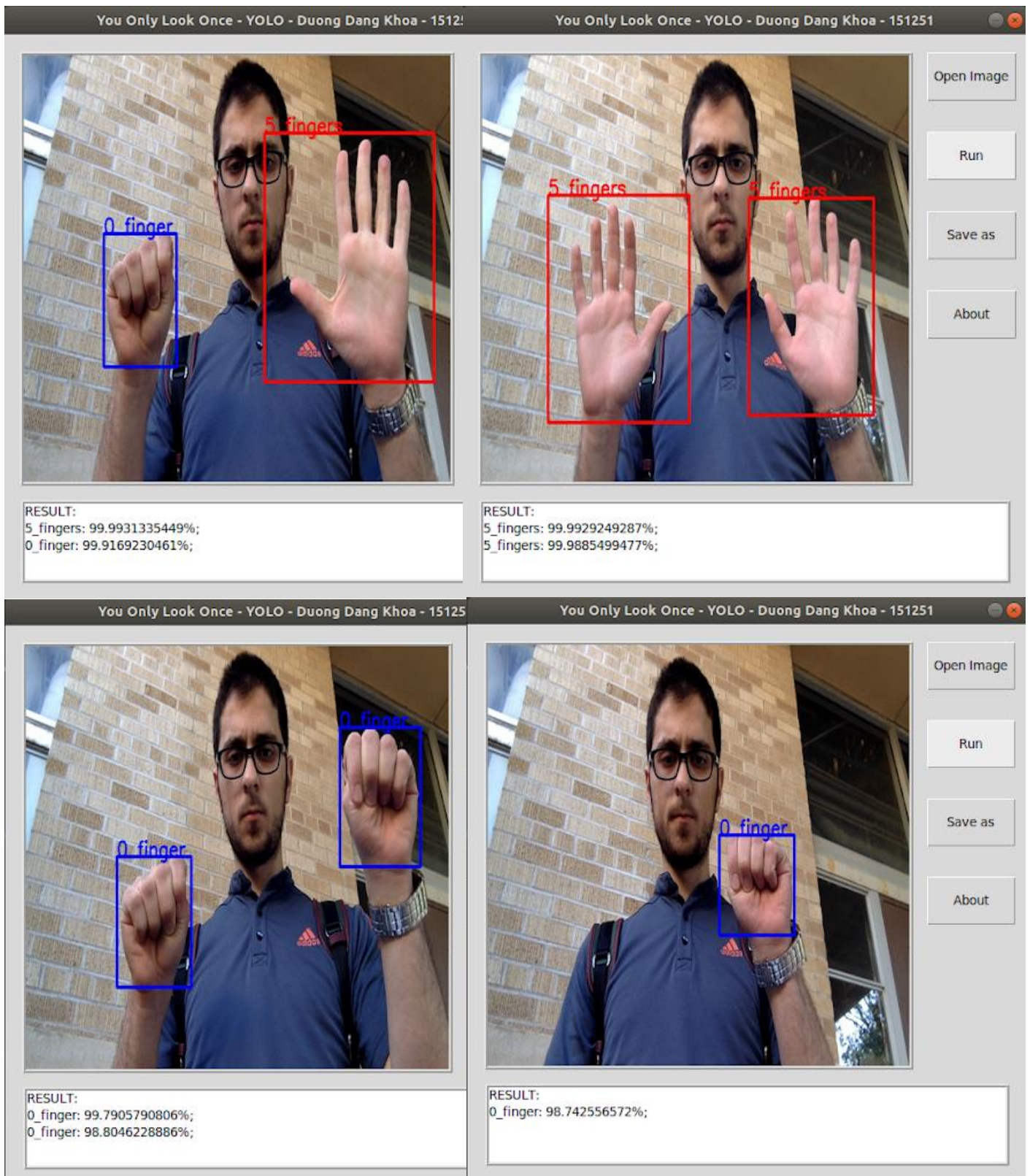
để có thể lưu file sau mỗi 100 vòng lặp và cho đến khi vòng lặp $i \geq 10000$ thì sẽ lưu theo 10000 vòng lặp 1 lần.

- Bước 10: Sau khi huấn luyện xong, em có vẽ biểu đồ để quyết định xem nên chọn bộ trọng số nào là tối ưu. Một bộ trọng số tối ưu khi có độ lỗi loss và avg_loss thấp. Dựa vào đồ thị ta thấy trọng số ở 110000 là tốt nhất nên em chọn bộ trọng số này.

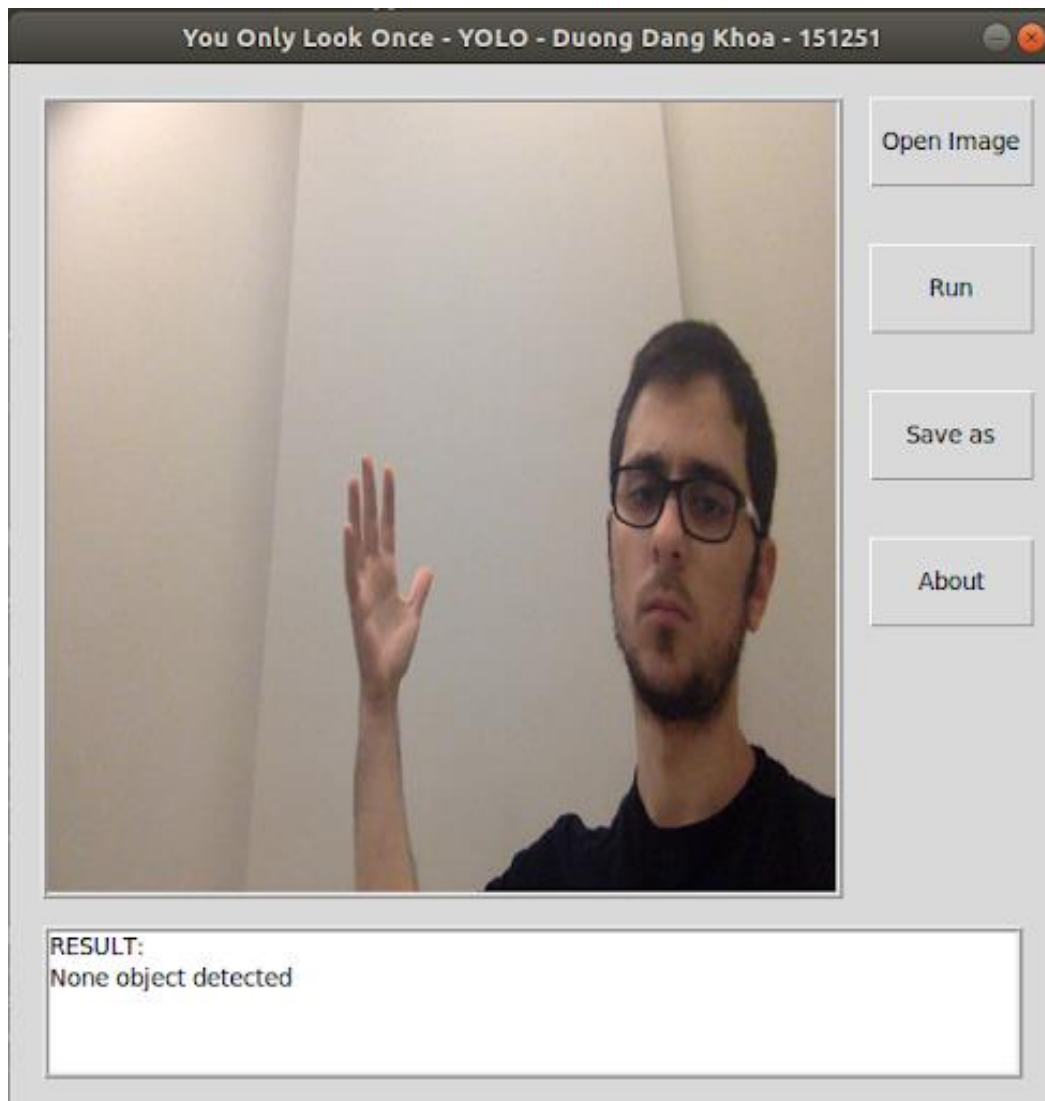


- ✓ Trong đồ án này, em đã train được 130000 vòng lặp nhưng do dữ liệu hình ảnh đưa vào khá phức tạp và khá ít nên độ chính xác chưa được cao đối với 1 số ảnh có độ sáng thấp hoặc góc độ hình ảnh khác với đa số ảnh còn lại. Dưới đây là kết quả thực thi yolo với file weights (custom_110000.weights) được train trên một số hình ảnh.

✓ Bên dưới là một số hình ảnh có độ chính xác cao



- ✓ Dưới đây là hình ảnh chưa detect được vì độ nghiêng của bàn tay



III/ Tài liệu tham khảo

- 1/ <https://pjreddie.com/darknet/yolo/>
- 2/ <https://towardsdatascience.com/yolo-you-only-look-once-real-time-object-detectionexplained-492dc9230006>
- 3/ <http://cv-tricks.com/object-detection/faster-r-cnn-yolo-ssd/>
<https://blog.paperspace.com/how-to-implement-a-yolo-v3-object-detector-fromscratch-in-pytorch-part-4/>
- 4/ https://medium.com/@manivannan_data/how-to-train-yolov2-to-detect-customobjects-9010df784f36
- 5/ <https://pjreddie.com/media/files/papers/YOLOv3.pdf>
- 6/ <http://cs231n.github.io/convolutional-networks/>