

**TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN  
ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH**

**KHOA KHOA HỌC MÁY TÍNH**



## **MÁY HỌC**

**Đề Tài: PHÁT HIỆN VƯỢT ĐÈN ĐỎ THÔNG QUA CAMERA GIÁM SÁT**

**GVHD: Lê Đình Duy**

**Phạm Nguyễn Trường An**

**Lớp: CS114.K21**

**Nhóm : Trần Doãn Thuyền**

**TP. HỒ CHÍ MINH, NGÀY 07 THÁNG 08 NĂM 2020**

## Table of Contents

Chương 1 : Giới thiệu bài toán: .....	3
1.1 TÌNH HÌNH HIỆN NAY: .....	3
1.2 MÔ TẢ BÀI TOÁN: .....	3
1.3 XÂY DỰNG HỆ THỐNG PHÁT HIỆN:.....	3
Chương 2 : Xử lý dữ liệu: .....	4
2.1 MỤC ĐÍCH THU THẬP DỮ LIỆU: .....	4
2.2 QUÁ TRÌNH THU THẬP DỮ LIỆU:.....	4
CHƯƠNG 3: PHƯƠNG PHÁP ĐỀ XUẤT.....	6
3.1 GIỚI THIỆU MẠNG YOLO: .....	6
3.1.1 Mạng Fully Convolutional Neural.....	7
3.2 Quay lại bước xử lý dữ liệu: .....	8
3.3 Darknet:.....	9
3.3.1 Chuẩn bị file cấu hình:.....	9
3.3.2 Huấn luyện mạng: .....	10
CHƯƠNG 4: THỰC NGHIỆM CHƯƠNG TRÌNH.....	15
4.1. CÀI ĐẶT: .....	15
4.2. QUÁ TRÌNH THỰC NGHIỆM: .....	15

## Chương 1 : Giới thiệu bài toán:

### 1.1 TÌNH HÌNH HIỆN NAY:

Hiện nay, tình hình tai nạn giao thông vẫn là vấn đề nhức nhối của xã hội, bởi vì không những các tai nạn giao thông gây nên cho ta các thiệt hại về của cải mà cả về tính mạng con người. Vậy khi đi xét về nguyên nhân dẫn tới các vụ tai nạn này - có thể là do yếu tố thời tiết; hay do tác động của hệ tầng giao thông và ý thức của người tham gia giao thông. Nhưng xét cho cùng, các vụ tai nạn nghiêm trọng xuất phát từ ý thức của người tham gia giao thông như việc vi phạm vượt đèn đỏ chiếm một số lượng không nhỏ.

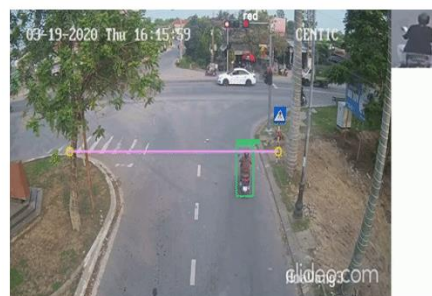
### 1.2 MÔ TẢ BÀI TOÁN:

Theo đó, trong đề tài này sẽ hướng tới việc xây dựng hệ thống phát hiện vi phạm vượt đèn đỏ bằng việc áp dụng các kĩ thuật Deep Learning trên dữ liệu thu được từ qua camera giám sát nhằm mục đích giảm một phần nào số lượng công việc cho các nhân viên giám sát và lực lượng chức năng ; và cũng như góp phần nhanh chóng phát hiện vi phạm và xử lý vi phạm triệt để và nhanh chóng nhằm góp phần nâng cao ý thức chung của người dân khi tham gia giao thông.

### 1.3 XÂY DỰNG HỆ THỐNG PHÁT HIỆN:



**Input: Video gốc.**



**Output: Video đã xác định vị trí đối tượng.**

Như vậy, bài toán phát hiện hành vi phạm vượt đèn đỏ là bài toán phức tạp bao gồm nhiều bài toán con như bài toán phát hiện (vehicle detection), bài toán phân loại tín hiệu (classification) và bài toán truy vết đối tượng (tracking) (hình 2 minh họa một số bài toán chọn trong hệ thống). Đây đều là các bài toán phức tạp trong lĩnh vực thị giác máy.

## Chương 2 : Xử lý dữ liệu:

### 2.1 MỤC ĐÍCH THU THẬP DỮ LIỆU:

Là một bước vô cùng quan trọng không thể bỏ qua khi bắt đầu một bài toán Học máy là thu thập dữ liệu để training cho model. Ví dụ đơn giản là các loại dữ liệu như hình ảnh, video, từ loại... Và vì thế, đa phần trong quá trình học Học Máy hay Thị Giác Máy Tính, chúng ta làm việc trên những bộ dữ liệu đã được xử lý sẵn. – hay được cung cấp từ các trang như Kaggle, ... . Nhưng trong thực tế, tùy thuộc vào bài toán mà cần thu thập dữ liệu khác nhau và ít khi tương quan với những bộ dữ liệu sẵn có.

Các model máy học không thể hoạt động nếu thiếu dữ liệu, trường hợp dữ liệu quá nhỏ thì dễ dẫn đến hiện tượng sự quá khớp (Overfitting) và model không thể học được đầy đủ các đặc trưng cho các trường hợp tổng quan hay nói một cách khác đi là model thiếu khả năng tổng quan hóa (Generalization) Vậy làm sao để tìm đủ dữ liệu huấn luyện và gắn nhãn cho nó? Có thể nói đây là công việc mất nhiều công sức nhất trong Học Máy

### 2.2 QUÁ TRÌNH THU THẬP DỮ LIỆU:

- Đi bộ: 372 đối tượng;
- Xe đạp: 15 đối tượng;
- Xe máy: 3511 đối tượng;
- Xe hàng rong: 3 đối tượng;
- Xe ba gác: 52 đối tượng;
- Xe taxi: 76 đối tượng;
- Xe hơi: 1026 đối tượng;
- Xe bán tải: 33 đối tượng;
- Xe cứu thương: 11 đối tượng;
- Xe khách: 141 đối tượng;
- Xe buýt: 194 đối tượng;
- Xe tải: 836 đối tượng;
- Xe container: 154 đối tượng;
- Xe cứu hỏa: 2 đối tượng

**Nhận xét về dữ liệu thu thập:**

Bộ dữ liệu bị skewed rất nhiều về đối tượng **xe máy** và **xe hơi** ; và rất ít đối với **xe cứu hỏa** và **xe hàng rong**

## CHƯƠNG 3: PHƯƠNG PHÁP ĐỀ XUẤT

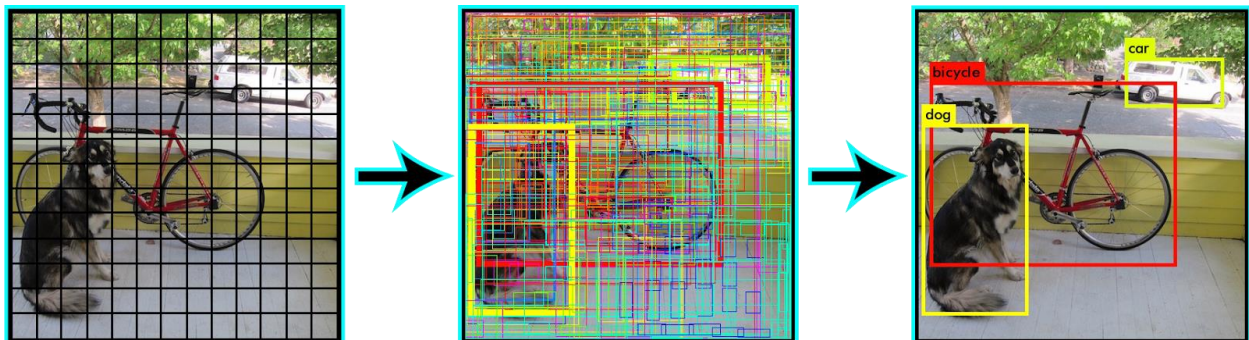
Nhóm sử dụng mạng yolo để phát hiện đối tượng trong video , rồi theo vết đối tượng trong video mạng các phương pháp tracking và với các rule ta đặt mềm bằng tay vì đối với từng video ta sẽ có từng vị trí đặt khác nhau.

### 3.1 GIỚI THIỆU MẠNG YOLO:

YOLO hay tên đầy đủ là **You only look once** là một mô hình CNN để phát hiện với một ưu điểm nổi trội là nhanh hơn nhiều so với những mô hình cũ như SSD.

Mặt khác, nó cũng tốt hơn các mạng có kiểu Region Proposal Classification network khác (ví dụ là Fast RCNN – và trong đây nhóm cũng có sử dụng mạng Faster RCNN để train cho mô hình – nhưng kết quả về mAP và mAR – mean average precision và average recall thấp nên sẽ không sử dụng vào hệ thống đề xuất).

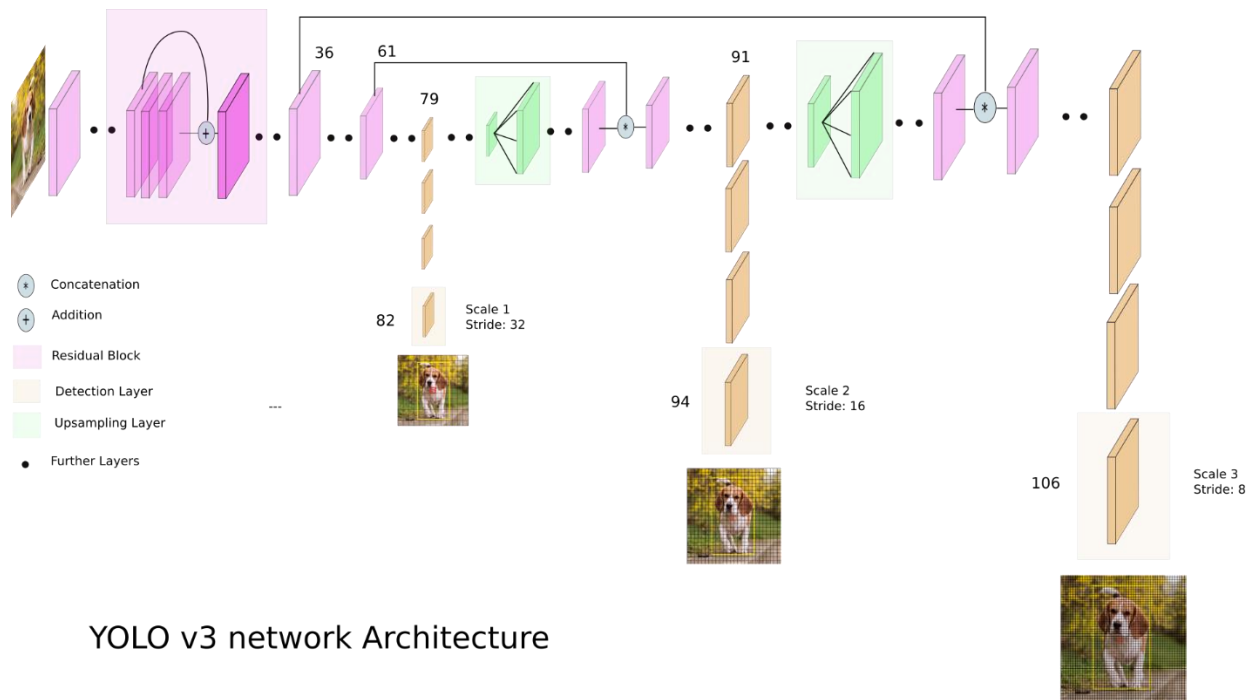
YOLO lại có một cách tiếp cận hoàn toàn khác, chỉ sử dụng duy nhất một neural network khi lấy tích chập trên bộ ảnh. Nghĩ cho cùng thì Kiến trúc của YOLO giống với FCNN (Full Convolution Neural Network) hơn, và hình ảnh chỉ được truyền qua FCNN một lần duy nhất, sau đó sẽ trả về output là các prediction. Hình ảnh đầu vào sẽ được chia thành các ô lưới (grid cell), và dự đoán các bounding box và xác suất phân loại cho mỗi grid cell. Các bounding box này được đánh trọng số kèm với nó là xác suất đã dự đoán.



Vậy chung quy lại ta sẽ có một output chuẩn là:

$$(pc, x, by, bh, bw, c)$$

Chiều dài sẽ là (5 + class)



### 3.1.1 Mạng Fully Convolutional Neural

	Type	Filters	Size	Output
1x	Convolutional	32	3 × 3	256 × 256
	Convolutional	64	3 × 3 / 2	128 × 128
	Convolutional	32	1 × 1	
	Convolutional	64	3 × 3	128 × 128
2x	Residual			128 × 128
	Convolutional	128	3 × 3 / 2	64 × 64
	Convolutional	64	1 × 1	
	Convolutional	128	3 × 3	64 × 64
8x	Residual			64 × 64
	Convolutional	256	3 × 3 / 2	32 × 32
	Convolutional	128	1 × 1	
	Convolutional	256	3 × 3	32 × 32
8x	Residual			32 × 32
	Convolutional	512	3 × 3 / 2	16 × 16
	Convolutional	256	1 × 1	
	Convolutional	512	3 × 3	16 × 16
4x	Residual			16 × 16
	Convolutional	1024	3 × 3 / 2	8 × 8
	Convolutional	512	1 × 1	
	Convolutional	1024	3 × 3	8 × 8
4x	Residual			8 × 8
	Residual			8 × 8
			Avgpool	Global
			Connected	1000
			Softmax	

Trong bài báo YOLO v3, các tác giả trình bày kiến trúc mạng mới, với nhiều lớp hơn 2 **kiến trúc rút trích đặc trưng** trước là **Darknet-53**. Và ý nghĩa cái con số 53, nó bao gồm 53 lớp tích chập, mỗi lớp theo sau là lớp chuẩn hóa (**batch normalization**) và dùng hàm kích hoạt **Leaky ReLU**. Để ý rằng là, trong kiến trúc này, tác giả **không lấy pool** và một lớp phức hợp, đồng thời tác giả dùng stride = 2 để giảm kích thước của **feature map**. Điều này tránh mất các đặc trưng cấp thấp (rút trích chưa được trừu tượng bởi các lớp chưa sâu của Darknet-53) khi ta gộp chung – vào các lớp phía sau (đã được trừu tượng hóa).

Vậy với input là 1 batch gồm các ảnh 416x416 thì đầu vào của yolo sẽ có chuẩn là (m, 416, 416, 3)

Sẽ có output tương ứng là:

$[(1, 13, 13, 255), (1, 26, 26, 255), (1, 52, 52, 255)]$

Tạm thời bỏ qua con số 255 vì nó chỉ là  $(80+5)*3$  theo công thức của darknet.

Còn 13,26,52 là kích thước gridcell.

object 1?	object 2?	offset x1	offset y1	width 1	height 1	offset x2	offset y2	width 2	height 2	0	0	1
-----------	-----------	-----------	-----------	---------	----------	-----------	-----------	---------	----------	---	---	---

### 3.2 Quay lại bước xử lý dữ liệu:

Vì ta cần làm theo một số chuẩn của darknet đề xuất vậy nên cần có một bước chuyển từ file .xml sang file .txt phù hợp cho việc train , evaluate của YOLO

Tạo yolo bounding box annotation : Ta sẽ tạo một .txt file ứng với mỗi ảnh .jpg cùng tên và đặt trong cùng 1 thư mục. Thông tin trong mỗi file .txt gồm có số lượng object và tọa độ của object ở trong ảnh, ứng với mỗi object là một dòng: `<object-class> <x> <y>`

`<width> <height>`

Ví dụ về file .txt : xe\_99.txt bao gồm các dòng như sau:

```
MobaTextEditor
File Edit Search View Format Syntax Special Tools
pascalvoc.py classes.txt xe_99.txt
1 2 0.68203125 0.7294600938967136 0.04635416666666667 0.2687793427230047
2 2 0.565625 0.24002347417840375 0.034375 0.16079812206572772
3 2 0.6244791666666667 0.2359154929577465 0.02604166666666667 0.14788732394366197
4 2 0.8645833333333334 0.49237089201877937 0.03854166666666667 0.20539906103286384
5 2 0.8559895833333333 0.3568075117370892 0.03802083333333333 0.1643192488262911
6 2 0.5854166666666667 0.1449530516431925 0.028125 0.13497652582159625
7 2 0.55546875 0.05046948356807512 0.02552083333333333 0.10093896713615023
8 0 0.8192708333333333 0.2717136150234742 0.028125 0.15375586854460094
9 0 0.91171875 0.4107981220657277 0.03177083333333333 0.20187793427230047
10 12 0.4096354166666666 0.1971830985915493 0.1390625 0.3943661971830986
11 11 0.2877604166666667 0.09917840375586855 0.10052083333333334 0.1983568075117371
12 6 0.078125 0.306924882629108 0.128125 0.20539906103286384
13
```

Note: Có khá nhiều tool để giúp tạo bbox annotation cho yolo tuy nhiên bạn có thể tham khảo 1 tool mình dùng ở đây, nó khá là tối giản và dễ sử dụng. Có 3 điều cần chú ý. Thứ nhất là tạo file classlist.txt (hay classes.txt , ... ) tùy theo từng tác giả đặt tên cho file



này ... nhưng không quan trọng cái tên mà quan trọng là nội dung bên trong phải gồm tên các class object, mỗi tên nằm trên 1 dòng.

### 3.3 Darknet:

Đây không phải là thuật ngữ để chỉ nơi chứa tất cả các dữ liệu đen tối còn lại của toàn bộ internet thế giới – mà ở đây là một công cụ vô cùng mạnh mẽ - một open-source code được viết bởi Redmon, Joseph and Farhadi và Ali

Open-source này được viết bằng ngôn ngữ C và CUDA (Compute Unified Device Architecture ) – một platform được viết nên để cho các hệ kiến trúc card đồ họa GPU – 100 đến 1000 cores ( dù tốc độ chậm hơn CPU thông thường – nhưng phù hợp tính toán song song).

Cùng với các requirements khác về darknet sẽ không được đề cập vì tranh chính của darknet sẽ chỉ dẫn hết.

#### 3.3.1 Chuẩn bị file cấu hình:

1. Tạo file `yolo-obj.cfg` có cùng nội dung với [yolov3.cfg](#) (đơn giản chỉ là copy) và paste :
  - Thay đổi dòng batch thành batch = 64 (hay 32 , 16 , ... )
    - Thường thì ta lấy các theo mũ của 2 : vì nó phù hợp với kiến trúc bên trong của card đồ họa;
    - **Note** : Nhưng nếu lấy quá thấp thì sẽ quá trình train sẽ rất lâu.
  - Thay đổi dòng subdivisions thành subdivision = 64 ( 32 , 16 ... ) nhưng phải là nhỏ hơn dòng batch . Dòng này có ý nghĩa rằng : nếu không may hết ram trên GPU thì ta sẽ càng chia nhỏ ra batch : lấy thành 16 hay 8 ... để tránh bị tràn ram – hỏng cả quá trình train trên Colab – ( một kinh nghiệm xương máu ... )
  - Thay `classes=80` thành số class object của bạn tại: [dòng 610](#), [dòng 696](#) và [dòng 783](#)
    - Trong đây thì thành 14 class.

- Thay `[filters=255]` thành `filters = (classes + 5)*3` tại: [dòng 603](#), [dòng 689](#) và [dòng 776](#). Ví dụ: nếu số class bằng 1 thì ta có `filters=18`, số class bằng 2 thì `filters=21`.

- Trong đây sẽ thành 57

**Note :** Hai dòng này phải đọc thật kĩ và chỉnh lại tham số cho đúng.

2. Tạo file `obj.names` với tên của mỗi class được đặt vào 1 dòng (giống với file `class_list.txt` bên trên)

3. Tạo file `obj.data` chứa:

```
classes= 14
train = data/train.txt
valid = data/test.txt
names = data/obj.names
backup = backup/
```

Trong đó:

File `train.txt` và `test.txt` chứa danh sách tên các ảnh, mỗi ảnh một dòng cùng với đường dẫn đến ảnh đó. Ví dụ:

### 3.3.2 Huấn luyện mạng:

Vào thư mục chứa darknet và bằng dòng lệnh:

`./darknet detector train data/obj.data yolo-obj.cfg darknet53.conv.74` ( đây là trên Linux nha) còn Window `darknet.exe detector train data/obj.data yolo-obj.cfg darknet53.conv.74`.

Lưu ý : vì ta đang train bằng file pretrained-darknet nên quá trình train sẽ rất lâu.

Trong bài viết gốc: darknet sẽ train – cứ 100 iteration sẽ lưu file weights một lần và cho đến 900. Rồi từ đó cứ 1000 iteration thì nó sẽ lưu file weights dưới 2 tên

- `yolo-obj_last.weights`; và
- `yolo-obj_xxxx.weights` cứ 1000 iterations.

... tất cả đều chứa trong folder backup.

Một kinh nghiệm của mình train là :

```
65         if(i%1000==0 || (i < 1000 && i%100 == 0)){
66             char buff[256];
67             sprintf(buff, "%s/%s_%d.weights", backup_directory, base, i);
68             save_weights(net, buff);
69         }
```

Chuyển thành if ( i% 200 == 0 ) {

...

}

Vì mình rất sợ việc đang train 1000 vòng mà đến iteration 900 colab disconnect giữa chừng, mất hết cả một quá trình huấn luyện.

Kết quả về độ đánh giá sau khi train:

552

```
detections_count = 8232, unique_truth_count = 6227
class_id = 0, name = di_bo , ap = 38.61% (TP = 195, FP = 177)
class_id = 1, name = xe_dap , ap = 58.00% (TP = 11, FP = 4)
class_id = 2, name = xe_may , ap = 66.84% (TP = 2473, FP = 1038)
class_id = 3, name = xe_hang_rong , ap = 100.00% (TP = 3, FP = 0)
class_id = 4, name = xe_ba_gac , ap = 82.69% (TP = 44, FP = 8)
class_id = 5, name = xe_taxi , ap = 82.23% (TP = 60, FP = 16)
class_id = 6, name = xe_hoi , ap = 80.50% (TP = 846, FP = 180)
class_id = 7, name = xe_ban_tai , ap = 96.47% (TP = 30, FP = 3)
class_id = 8, name = xe_cuu_thuong , ap = 65.57% (TP = 7, FP = 4)
class_id = 9, name = xe_khach , ap = 80.43% (TP = 116, FP = 25)
class_id = 10, name = xe_buyt , ap = 94.05% (TP = 185, FP = 9)
class_id = 11, name = xe_tai , ap = 92.44% (TP = 762, FP = 74)
class_id = 12, name = xe_container , ap = 97.95% (TP = 146, FP = 8)
class_id = 13, name = xe_cuu_hoa , ap = 100.00% (TP = 2, FP = 0)
```

Với việc lấy ngưỡng chính xác là 0.25:

precision = 0.76, recall = 0.78, F1-score = 0.77 và

TP = 4880, FP = 1546 và average IoU = 64.57 %

Với việc lấy ngưỡng IOU là 0.75 theo phương pháp đánh giá COCO:

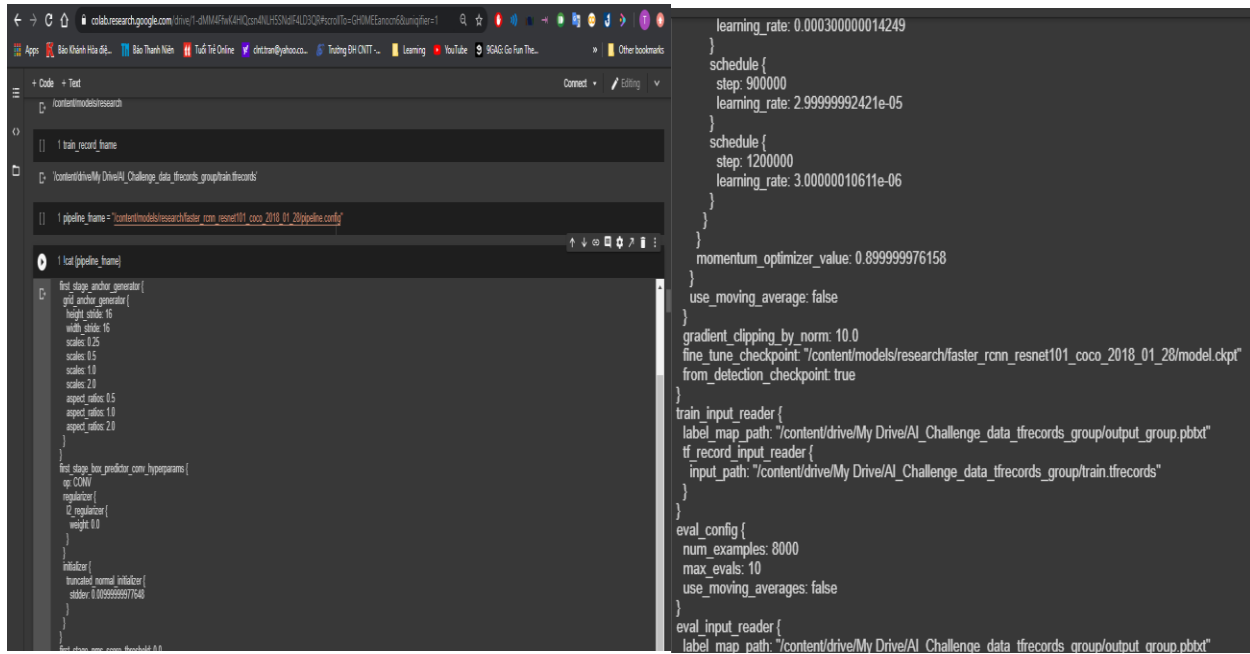
Used Area-Under-Curve mean average precision sẽ là (mAP@0.75) = 81.13 %

IoU threshold = 75 %, used Area-Under-Curve for each unique Recall  
mean average precision (mAP@0.75) = 0.811269, or 81.13 %

Nhóm cũng có sử dụng mạng Faster-RCNN – train trên API do tensorflow cung cấp, và khác với darknet ... tensorflow api cần file tfrecord

```
[ ] 1 # NOTE: Update these TFRecord names from "cells" and "cells_label_map" to your files!
    2 test_record_fname = '/content/drive/My Drive/AI_Challenge_data_tfrecords_group/eval.tfrecords'
    3 train_record_fname = '/content/drive/My Drive/AI_Challenge_data_tfrecords_group/train.tfrecords'
    4 label_map_pbtxt_fname = '/content/drive/My Drive/AI_Challenge_data_tfrecords_group/output_group.pbtxt'
```

Và file nó cần là pipe\_line : khác với file config của darknet-yolo



```
learning_rate: 0.000300000014249
}
schedule {
  step: 900000
  learning_rate: 2.99999992421e-05
}
schedule {
  step: 1200000
  learning_rate: 3.00000010611e-06
}
momentum_optimizer_value: 0.899999976158
}
use_moving_average: false
}
gradient_clipping_by_norm: 10.0
fine_tune_checkpoint: "/content/models/research/faster_rcnn_resnet101_coco_2018_01_28/model.ckpt"
from_detection_checkpoint: true
}
train_input_reader {
  label_map_path: "/content/drive/My Drive/AI_Challenge_data_tfrecords_group/output_group.pbtxt"
  tf_record_input_reader {
    input_path: "/content/drive/My Drive/AI_Challenge_data_tfrecords_group/train.tfrecords"
  }
}
eval_config {
  num_examples: 8000
  max_evals: 10
  use_moving_averages: false
}
eval_input_reader {
  label_map_path: "/content/drive/My Drive/AI_Challenge_data_tfrecords_group/output_group.pbtxt"
```

## Cài đặt môi trường và huấn luyện mạng

```
1 %cd /content
2 !git clone --quiet https://github.com/tensorflow/models.git
3 !apt-get install -qq protobuf-compiler python-pil python-bbox
4 !pip install -q Cython contextlib2 pillow bml matplotlib
5 !pip install -q pycocotools
6 %cd /content/models/research
7 !protoc object_detection/protos/*.proto --python_out=.
8 import os
9 os.environ["PYTHONPATH"] += "/content/models/research/slim/

/content
Selecting previously unselected package python-bs4.
(Reading database ... 144487 files and directories currently installed.)
Preparing to unpack .../0-python-bs4_4.6.0-1_all.deb ...
Unpacking python-bs4 (4.6.0-1) ...
Selecting previously unselected package python-pkg-resources.
Preparing to unpack .../1-python-pkg-resources_39.0.1-2_all.deb ...
Unpacking python-pkg-resources (39.0.1-2) ...
Selecting previously unselected package python-chardet.
Preparing to unpack .../2-python-chardet_3.0.4-1_all.deb ...
Unpacking python-chardet (3.0.4-1) ...

1 !python "/content/drive/My Drive/models/research/object_detection/model_main.py" \
2 --pipeline_config_path="/content/drive/My Drive/Faster RCNN/training/pipeline.config" \
3 --model_dir="/content/drive/My Drive/Faster RCNN/training" \
4 --alsologtostderr \
5 --num_train_steps=250000 \
6 --num_eval_steps=100

2020-08-12 04:31:38.036520 I tensorflow/compiler/xla/service/service.cc:168] XLA service 0x131149c0 initialized for platform CUDA (this does not guarantee that XLA
2020-08-12 04:31:38.036562 I tensorflow/compiler/xla/service/service.cc:176] StreamExecutor device (0): Tesla T4, Compute Capability 7.5
2020-08-12 04:31:38.036763 I tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:983] successful NUMA node read from SysFS had negative value (-1), but the
2020-08-12 04:31:38.037297 I tensorflow/core/common_runtime/gpu/gpu_device.cc:1618] Found device 0 with properties:
name: Tesla T4 major: 7 minor: 5 memoryClockRate(GHz): 1.59
pciBusID: 0000:00:04:0
2020-08-12 04:31:38.049536 I tensorflow/stream_executor/platform/default/dso_loader.cc:44] Successfully opened dynamic library libcudart.so.10.0
2020-08-12 04:31:38.224262 I tensorflow/stream_executor/platform/default/dso_loader.cc:44] Successfully opened dynamic library libcublas.so.10.0
2020-08-12 04:31:38.315665 I tensorflow/stream_executor/platform/default/dso_loader.cc:44] Successfully opened dynamic library libcufft.so.10.0
2020-08-12 04:31:38.334564 I tensorflow/stream_executor/platform/default/dso_loader.cc:44] Successfully opened dynamic library libcurand.so.10.0
2020-08-12 04:31:38.581560 I tensorflow/stream_executor/platform/default/dso_loader.cc:44] Successfully opened dynamic library libcusolver.so.10.0
2020-08-12 04:31:38.708305 I tensorflow/stream_executor/platform/default/dso_loader.cc:44] Successfully opened dynamic library libcusparse.so.10.0
2020-08-12 04:31:39.213464 I tensorflow/stream_executor/platform/default/dso_loader.cc:44] Successfully opened dynamic library libcudnn.so.7
2020-08-12 04:31:39.213697 I tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:983] successful NUMA node read from SysFS had negative value (-1), but the
2020-08-12 04:31:39.214325 I tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:983] successful NUMA node read from SysFS had negative value (-1), but the
2020-08-12 04:31:39.216200 I tensorflow/core/common_runtime/gpu/gpu_device.cc:1159] Device interconnect StreamExecutor with strength 1 edge matrix:
2020-08-12 04:31:39.216232 I tensorflow/core/common_runtime/gpu/gpu_device.cc:1165] 0
2020-08-12 04:31:39.216242 I tensorflow/core/common_runtime/gpu/gpu_device.cc:1178] 0: N
2020-08-12 04:31:39.216395 I tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:983] successful NUMA node read from SysFS had negative value (-1), but the
2020-08-12 04:31:39.216964 I tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:983] successful NUMA node read from SysFS had negative value (-1), but the
2020-08-12 04:31:39.217892 I tensorflow/core/common_runtime/gpu/gpu_device.cc:391] Overriding allow_growth setting because the TF_FORCE_GPU_ALLOW_GROWTH environment variable is set.
2020-08-12 04:31:39.217646 I tensorflow/core/common_runtime/gpu/gpu_device.cc:1204] Created TensorFlow device (job:localhost/replica:0/task:0/device:GPU:0 w
INFO:tensorflow:Restoring parameters from /content/drive/My Drive/Faster RCNN/training/model_checkpoint_217892
I0812 04:31:39.221922 139827639433088 saver.py:1284] Restoring parameters from /content/drive/My Drive/Faster RCNN/training/model_checkpoint_217892
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/tensorflow_core/python/training/saver.py:1069: get_checkpoint_mtimes (from tensorflow.python.train
Instructions for updating:
Use standard file utilities to get mtimes.
```

## Kết quả đánh giá:

```
Average Precision (AP) @[ IoU=0.50:0.95 | area= all | maxDets=100 ] = 0.037
Average Precision (AP) @[ IoU=0.50 | area= all | maxDets=100 ] = 0.075
Average Precision (AP) @[ IoU=0.75 | area= all | maxDets=100 ] = 0.031
Average Precision (AP) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.000
Average Precision (AP) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.016
Average Precision (AP) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.051
Average Recall (AR) @[ IoU=0.50:0.95 | area= all | maxDets= 1 ] = 0.068
Average Recall (AR) @[ IoU=0.50:0.95 | area= all | maxDets= 10 ] = 0.277
Average Recall (AR) @[ IoU=0.50:0.95 | area= all | maxDets=100 ] = 0.288
Average Recall (AR) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.001
Average Recall (AR) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.147
Average Recall (AR) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.354
```

Các chỉ số average precision và average recall đều thấp hơn của YOLO

## CHƯƠNG 4: THỰC NGHIỆM CHƯƠNG TRÌNH

### 4.1. CÀI ĐẶT:

Ngôn ngữ sử dụng: Ngôn ngữ Python 3.7

Công cụ thực hiện: Google Colab

Cấu hình máy thực hiện: Tesla P100 16GB, 25gb RAM – cung cấp bởi Colab

Thư viện sử dụng: darknet, tensorflow, scipy, scikit-learn, opencv-python, h5py, matplotlib, pillow, requests, psutil, flask, pandas, numpy, login, pickle.

### 4.2. QUÁ TRÌNH THỰC NGHIỆM:

# Thiết lập file weights và config

```
weights_path = os.path.sep.join([args["yolo"], "yolov3.weights"])
```

```
config_path = os.path.sep.join([args["yolo"], "yolov3.cfg"])
```

# Load vào file weight và lấy ra 3 lớp output như đã nói như trên

```
print("[INFO] loading YOLO from disk...")
```

```
net = cv2.dnn.readNetFromDarknet(config_path, weights_path)
```

```
ln = net.getLayerNames()
```

```
ln = [ln[i[0] - 1] for i in net.getUnconnectedOutLayers()]
```

# Khởi tạo nguồn stream từ video đầu vào, khởi tạo file ghi writer

```
vs = cv2.VideoCapture(args["input"])
```

```
fps = vs.get(cv2.CAP_PROP_FPS)
```

```
writer = None
```

```
writer2 = None
```

```
(W, H) = (None, None)
```

# Khởi tạo các yếu tố như frame rate

try:

```
prop = cv2.cv.CV_CAP_PROP_FRAME_COUNT if imutils.is_cv2() \
    else cv2.CAP_PROP_FRAME_COUNT
total = int(vs.get(prop))
print("[INFO] {} total frames in video".format(total))
```

# Chống lỗi khi người dùng nhập sai input

except:

```
print("[INFO] could not determine # of frames in video")
print("[INFO] no approx. completion time can be provided")
total = -1
```

# Để tất cả các label của các bounding box ... đều có màu như nhau ...

```
np.random.seed(42)
```

# Khởi tạo file label:

```
labelsPath = os.path.sep.join([args["yolo"], "coco.names"])
LABELS = open(labelsPath).read().strip().split("\n")
COLORS = np.random.randint(0, 255, size=(len(LABELS), 3),
    dtype="uint8")
print("Video in:",args["input"])
print("Video out:",args["output"])
```

```
VIDEO_INFO = read_video_log(args["input"])
```

```
print(VIDEO_INFO)
```

```
NEW_VIDEO_INFO = []
```



# Đọc từ video cũ : vì nếu đã xử lý video thì không cần phải dẫn tạo rule lại cho việc phát hiện đối tượng:

```
if str(type(VIDEO_INFO)) == "<class 'NoneType'>":
    NEW_VIDEO_INFO.append(args['input'])
    NEW_VIDEO_INFO.append(args['output'])
    list_coors = setLightCoordinates(args,net,ln)
    print("Select Coordinates:")
    select_index = int(input())
    xlight,ylight,wlight,hlight = list_coors[select_index]
    red_light_coors = (xlight,ylight,wlight,hlight)
    for coor in red_light_coors:
        NEW_VIDEO_INFO.append(coor)
    print("Select threshold red light:")
    threshold_red_light = int(input())
    NEW_VIDEO_INFO.append(threshold_red_light)
    # Detection boundary: from top left to bottom right:
    print("Input Detection rule:")
    print("Top left coordinates:")
    tl = (int(input()),int(input()))
    print("Bottom right coordinates:")
    br = (int(input()),int(input()))
    NEW_VIDEO_INFO.append(tl[0])
    NEW_VIDEO_INFO.append(tl[1])
    NEW_VIDEO_INFO.append(br[0])
    NEW_VIDEO_INFO.append(br[1])
```

```

write_video_log(NEW_VIDEO_INFO)

# Còn nếu không thì khởi tạo Rule như thường
else:
    xlight,ylight,wlight,hlight = VIDEO_INFO[2:6]
    threshold_red_light = VIDEO_INFO[6]
    tl = VIDEO_INFO[7:9]
    br = VIDEO_INFO[9:11]

detection_rule = DetectionRule(tl,br)

ctr = 0
vehicle_list = []
# khởi tạo tracker:
def init_tracker(img,box):
    tracker = cv2.TrackerCSRT_create()
    (x, y, w, h) = [int(v) for v in box[0:4]]
    classID = box[4]
    success = tracker.init(img, (x, y, w, h))
    vehicle = Vehicle(x, y, w, h,classID,ctr,tracker)
    vehicle_list.append(vehicle)
# Hàm phụ trợ cho Tracker:
def update_tracker(img,color):
    for i, vehicle in enumerate(vehicle_list):
        ok = vehicle.update_box(img)
        if color == 'Xanh':

```

```

        vehicle.update_last_green_light_pos()
    else:
        if vehicle.middle_point[1] < threshold_red_light - 30: #Soft margin
            if vehicle.green_light_last_pos[1] > threshold_red_light:
                vehicle.update_violation()
            else :
                pass

detect_count = 0
# Tạo stack các khung rỗng để chứa đối tượng bị vi phạm
recent_violation_crop_img = [np.ones((400,400,3),dtype=np.uint8)*255 for x in
range(5)]
a = np.vstack(np.ones((400,400,3),dtype=np.uint8)*255 for x in range(5))
# Hàm phụ trợ giúp ghi text tiếng Việt ... (khi làm cái này bị rất nhiều lỗi font nên ta cần
file arial.ttf .
def np_PIL_np(image,x,y,color,text):
    PIL_image = Image.fromarray(np.uint8(image)).convert('RGB')
    draw = ImageDraw.Draw(PIL_image)
    draw.text((x, y - 10),text,fill = tuple(color),font=font)
    image = np.array(PIL_image)
    return image
# Vẽ bounding cho đối tượng:
def draw_bounding(img,vehicle_list):
    global detect_count
    global a
    global recent_violation_crop_img

```

```

original_img = img.copy()
p1 = (detection_rule.tl[0], threshold_red_light)
p2 = (detection_rule.br[0], threshold_red_light)
# (255, 153, 255) Light Blue
# (0,255,255) Yellow
cv2.line(img, p1, p2, (255, 153, 255), 3, cv2.LINE_AA)
cv2.circle(img,p1,10,(0,255,255),2)
cv2.circle(img,p2,10,(0,255,255),2)
stack = None

```

```

for i, vehicle in enumerate(vehicle_list):
    x1,y1,x2,y2 = vehicle.coors_to_bounding_box()

    # img_pad = vehicle.draw_track_line(img_pad)

    color = [int(c) for c in COLORS[vehicle.classID]]
    cv2.rectangle(img, (x1, y1), (x2, y2), color, 2)

    text = "{}".format(LABELS[vehicle.classID])

    img = np_PIL_np(img,x1,y1,color,text)
    if vehicle.recent_violated == True:
        if vehicle.drawn_counting_frame != 0:
            cv2.rectangle(img,(x1,y1),(x2,y2),(0, 0, 255), 3)

```

```

        vehicle.drawn_counting_frame -= 1
    if vehicle.take_picture == False:
        cv2.rectangle(img,(x1,y1),(x2,y2),(0, 0, 255), 3)
        detect_path = "detect/"+args["input"].split("/")[-1].replace(".mp4","") + "-"
+str(detect_count) + ".jpg"
        detect_count +=1
        cv2.imwrite(detect_path,img)
        vehicle.take_picture = True
        a,recent_violation_crop_img = side_way_image(original_img,
            (x1,x2,y1,y2),recent_violation_crop_img)

    img_pad = np.hstack((cv2.resize(img,(3000,2000)),a))
    img_pad = cv2.resize(img_pad,(img.shape[1],img.shape[0]))
    # try:
    #     text = "Xe vi ph?m"
    #     y,x = original_img.shape[:2]
    #     cv2.putText(img_pad, text, (x, y + 5),
    #         cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 0, 255), 2)
    # except:
    #     pass
    return img_pad

#Processing time
startTime = time.time()

### Vào phần chính của chương trình
#

```

```

while True:
    (grabbed, frame) = vs.read()
    if(not grabbed ):
        break

    # Trường hợp ta chưa lấy kích thước khung hình
    if W is None or H is None:
        (H, W) = frame.shape[:2]

    # Đây là quá trình hậu xử lý các đối tượng trong vehicle_list :
    # nếu quá nhiều sẽ làm giảm tốc độ xử lý một frame
    if ctr%15 == 0:
        vehicle_list = clean_vehicle_list(vehicle_list, ctr,detection_rule)
        print("Current tracking:",len(vehicle_list),"vehicle")

    frame_temp = frame.copy()
    light = frame[ylight:ylight + hlight, xlight:xlight + wlight]
    b, g, r = cv2.split(light)
    light = cv2.merge([r, g, b])
    color = trafficLightColor.estimate_label(light)
    update_tracker(frame_temp,color)
    #

    frame_temp = np_PIL_np(frame_temp,xlight,ylight,(255, 255,
255),trafficLightColor.estimate_label(light))
    # Phát hiện đối tượng tất cả các khung hình rất tốn chi phí vì thế cứ 5 frame ta
    # mới detect 1 lần:

```

```

if(ctr % 5== 0):
    #
    if not grabbed:
        break

    # Tạo blob – một đối tượng để đưa qua mạng detect và cho ra kết quả là các
    bounding box

    # Kèm theo xác suất chính xác của đối tượng mà YOLO dự đoán trong khung hình

    blob = cv2.dnn.blobFromImage(frame, 1 / 255.0, (416, 416),
        swapRB=True, crop=False)
    net.setInput(blob)
    start = time.time()
    layerOutputs = net.forward(ln)
    end = time.time()

    print("Frame ",ctr," take ",end-start," seconds!")

    # Xử lý output của yolo theo mẫu ra bounding box , độ chính xác và class id – trả về
    bằng dictionary.

    output = yolo_output(layerOutputs, args["confidence"],image_W = W, image_H= H
        ,detection_rule = detection_rule ,classID_list=classID_list)

    # Khởi tạo lists các bounding box, độ chính xác và class id tương ứng.

    frame_boxes = output["Frame boxes"]
    confidences = output["Confidences"]
    classIDs = output["Class IDs"]

```

```

# Hàm non-maxima suppression để loại bỏ các bounding box có độ chính xác thấp
idxs = cv2.dnn.NMSBoxes(frame_boxes, confidences, args["confidence"],
                        args["threshold"])

currentBoxes = []
# Chắc chắn rằng 1 đối tượng trong đó được phát hiện
if len(idxs) > 0:
    # loop over the indexes we are keeping
    for i in idxs.flatten():
        # extract the bounding box coordinates
        (x, y) = (frame_boxes[i][0], frame_boxes[i][1])
        (w, h) = (frame_boxes[i][2], frame_boxes[i][3])
        classID = classIDs[i]
        con = confidences[i]
        currentBoxes.append((x,y,w,h,classID))

take_indx = []
# Xử lý cho việc intersect bounding box với các bounding box của các tracker
for i, box in enumerate(currentBoxes):
    not_take_flag = False
    if len(vehicle_list) == 0:
        take_indx.append(int(i))
    else:
        (x, y, w, h) = [int(v) for v in box[0:4]]
        object_bb = rect_to_bounding_box(x,y,w,h)
        for k, vehicle in enumerate(vehicle_list):

```



```

        vehicle_bb = vehicle.coors_to_bounding_box()
        val = bb_intersection_over_union(vehicle_bb,object_bb)
        if val > 0.1 and box[4] == vehicle.classID:
            not_take_flag = True
            break
        if not_take_flag == False:
            take_indx.append(i)
            init_tracker(frame_temp,currentBoxes[i])
## Hậu xử lý : vẽ và ghi khung hình
# Vẽ lên khung hình
    frame_temp = draw_bounding(frame_temp,vehicle_list)
# Ghi khung hình vào file
    if writer is None:
        # initialize our video writer

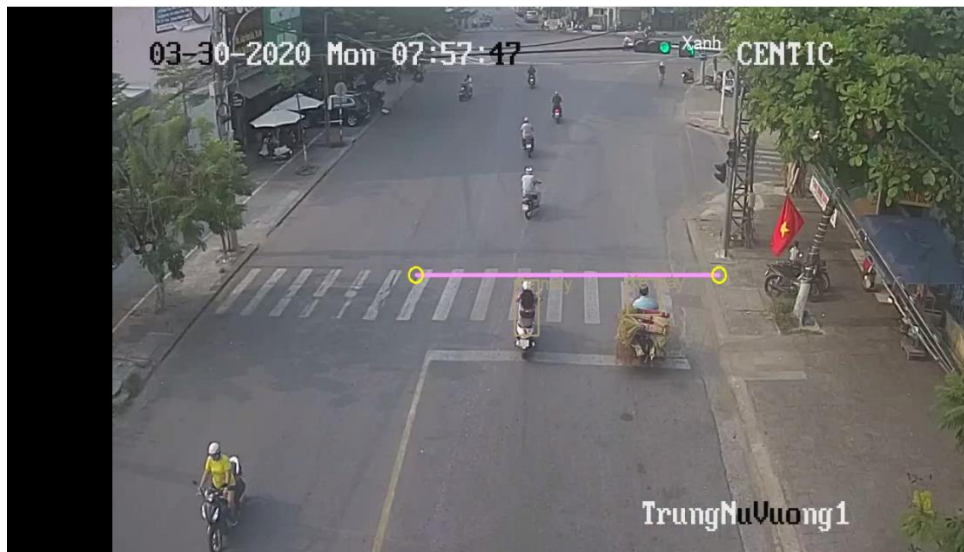
        fourcc = cv2.VideoWriter_fourcc(*"mp4v")
        writer = cv2.VideoWriter(args["output"], fourcc, fps
                                ,(frame_temp.shape[1], frame_temp.shape[0]), True)
    # some information on processing single frame
    if total > 0:
        elap = (end - start)
        print("[INFO] single frame took {:.4f} seconds".format(elap))
        print("[INFO] estimated total time to finish: {:.4f}".format(
            elap * total))
    writer.write(frame_temp)
    ctr = ctr + 1

```

```
# print(ctr)
# Giải phóng tài nguyên
# release the file pointers
print("[INFO] cleaning up...")
writer.release()
vs.release()
endTime = time.time()
print('Total Time: ', endTime - startTime)
```

## Kết quả thực nghiệm:

### 1. Khi đèn xanh:



Cả 2 đối tượng đều không bị ghi lên màn hình bên phải.

2. Khi đèn đỏ:



Xe thứ nhất vi phạm bị đưa vào khung hình bên cạnh.



**Xe thứ hai vi phạm bị đưa vào khung hình bên cạnh đưa lên trên hình xe thứ nhất vi phạm.**