	VIETTEL AI RACE	TD048
	NHẬN DIỆN VỊ TRÍ BIỂN SỐ XE MÁY VIỆT NAM	Lần ban hành: 1

1. Lời mở đầu

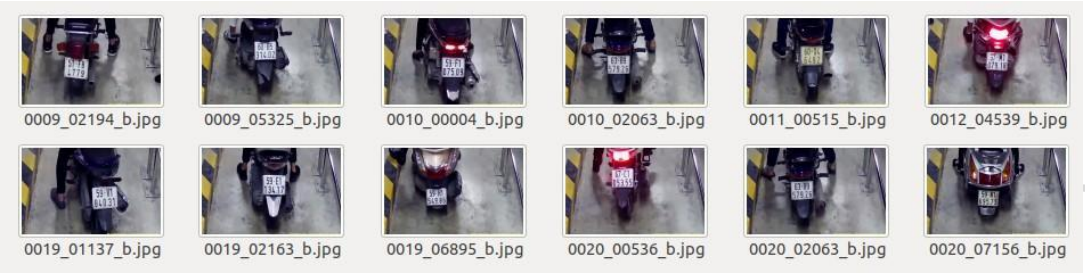
Bài toán nhận diện biển số xe Việt Nam là một bài toán không còn mới, đã được phát triển dựa trên các phương pháp xử lý ảnh truyền thống và cả những kỹ thuật mới sử dụng Deep Learning. Trong bài toán này tôi chỉ phát triển bài toán phát hiện biển số (một phần trong bài toán nhận diện biển số) dựa trên thuật toán YOLO-Tinyv4 với mục đích:

- Hướng dẫn chuẩn bị dữ liệu cho bài toán Object Detection.
- Hướng dẫn huấn luyện YOLO-TinyV4 dùng darknet trên Google Colab.

2. Chuẩn bị dữ liệu

2.1 Đánh giá bộ dữ liệu

Trong bài viết tôi sử dụng bộ dữ liệu biển số xe máy Việt Nam chứa 1750 ảnh, bạn đọc có thể tải tại đây.



Hình 14.1: Ảnh biển số trong bộ dữ liệu

Ảnh biển số xe được trong bộ dữ liệu được chụp từ một camera tại vị trí kiểm soát xe ra vào trong hầm. Do vậy:

- Kích thước các biển số xe không có sự đa dạng, do khoảng cách từ camera đến biển số xe xấp xỉ gần bằng nhau giữa các ảnh.
- Ảnh có độ sáng thấp và gần giống nhau do ảnh được chụp trong hầm chung cư.

=> Cần làm đa dạng bộ dữ liệu.

2.2 Các phương pháp tăng sự đa dạng của bộ dữ liệu

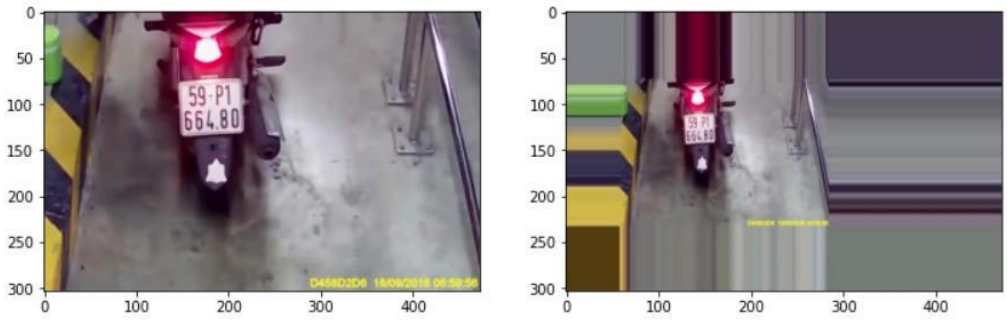
2.2.1 Đa dạng kích thước của biển số

Đa dạng kích thước bằng 2 cách:

- Cách 1: Thu nhỏ kích thước biển bằng cách thêm biên kích thước ngẫu nhiên vào ảnh gốc, sau đó resize ảnh bằng kích thước ảnh ban đầu.
- Cách 2: Crop ảnh chứa biển số với kích thước ngẫu nhiên, sau đó resize ảnh bằng kích thước ảnh ban đầu.

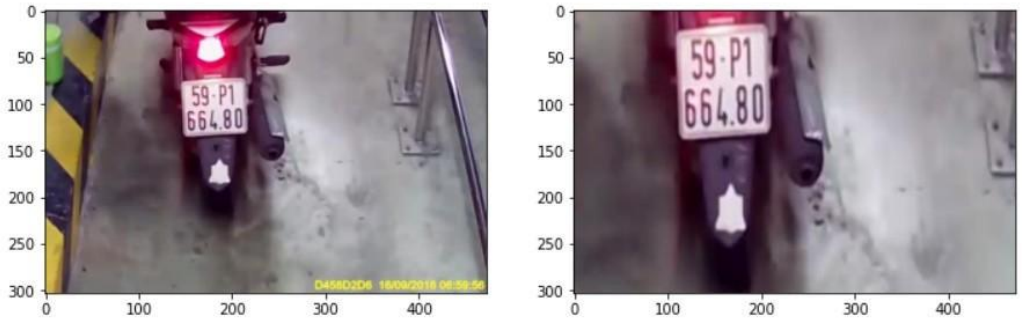
```
# Cách1 def add_boder(image_path, output_path, low, high):
""" low: kích thước biên thấp nhất (pixel) hight: kích thước biên lớn nhất (pixel)
"""
# random các kích thước biên trong khoảng (low, high)
```

```
top = random.randint(low, high)
bottom = random.randint(low, high)
left = random.randint(low, high)
right = random.randint(low, high)
image = cv2.imread(image_path)
original_width, original_height = image.shape[1], image.shape[0]
#sử dụng hàm của opencv để thêm biên
image = cv2.copyMakeBorder(image, top, bottom, left, right, cv2.BORDER_REPLICATE)
#sau đó resize ảnh bằng kích thước ban đầu của ảnh
image = cv2.resize(image, (original_width, original_height))
cv2.imwrite(output_path, image)
```



Hình 14.2: Ảnh thu được (bên phải) sau khi chạy hàm trên

```
# Cách2 def random_crop(image_path, out_path):
image = cv2.imread(image_path)
original_width, original_height = image.shape[1], image.shape[0]
x_center,y_center = original_height//2, original_width//2
x_left = random.randint(0, x_center//2)
x_right = random.randint(original_width-x_center//2, original_width)
y_top = random.randint(0, y_center//2)
y_bottom = random.randint(original_height-y_center//2, original_width)
# crop ra vùng ảnh với kích thước ngẫu nhiên
cropped_image = image[y_top:y_bottom, x_left:x_right]
# resize ảnh bằng kích thước ảnh ban đầu
cropped_image = cv2.resize(cropped_image, (original_width, original_height))
cv2.imwrite(out_path, cropped_image)
```



Hình 14.3: Ảnh thu được (bên phải) sau khi chạy hàm trên

2.2.2 Thay đổi độ sáng của ảnh

```
def change_brightness(image_path, output_path, value):
""" value: độ sáng thay đổi"""
img=cv2.imread(image_path)
hsv = cv2.cvtColor(img, cv2.COLOR_BGR2HSV)
h, s, v = cv2.split(hsv)
v = cv2.add(v, value)
v[v > 255] = 255
v[v < 0] = 0
final_hsv = cv2.merge((h, s, v))
img = cv2.cvtColor(final_hsv, cv2.COLOR_HSV2BGR)
cv2.imwrite(output_path, img)
```

Hình 14.4: Độ sáng thay đổi (bên phải)

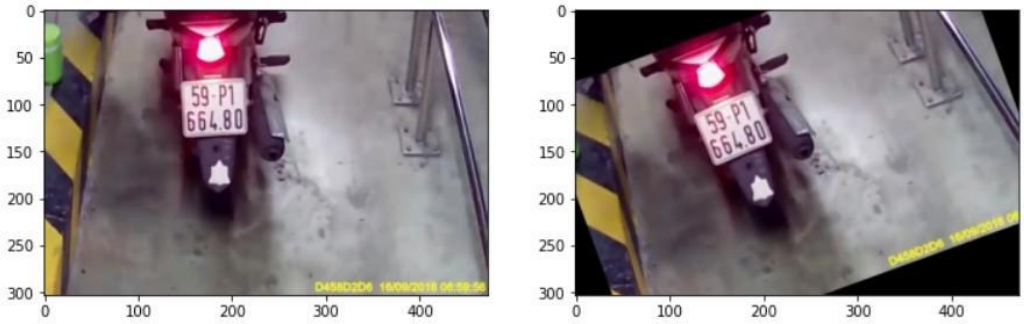
2.2.3 Xoay ảnh

```
import imutils def rotate_image(image_path, range_angle, output_path):
""" range_angle: Khoảng góc quay"""
image = cv2.imread(image_path)
#lựa chọn ngẫu nhiên góc quay
```

angle = random.randint(-range_angle, range_angle)

img_rot = imutils.rotate(image, angle)

cv2.imwrite(output_path, img_rot)



Hình 14.5: Ảnh được xoay (bên phải)

2.3 Gán nhãn dữ liệu

Tool gán nhãn ở đây tôi dùng là **labelImg**, bạn đọc có thể tải và đọc hướng dẫn sử dụng tại đây.

Hình 14.6: Xác định vùng biển chứa biển số

LabelImg hỗ trợ gán nhãn trên cả 2 định dạng PASCAL VOC và YOLO với phần mở rộng file annotation tương ứng là .xml và .txt.

Trong bài toán sử dụng mô hình YOLO, tôi lưu file annotation dưới dạng .txt.

```
0 0.384534 0.346535 0.201271 0.250825
```

Hình 14.7: Nội dung trong một file annotation

Mỗi dòng trong một file annotation bao gồm: <object-class> <x> <y> <width> <height>.

Trong đó: <x> <y> <width> <height> tương ứng là tọa độ trung tâm và kích thước của đối tượng. Các giá trị này đã được chuẩn hóa lại, do vậy giá trị luôn nằm trong đoạn [0,1]. object-class là chỉ số đánh dấu các classes.

Lưu ý: Với bài toán có nhiều nhãn, nhiều người cùng gán nhãn thì cần thống nhất với nhau trước về thứ tự nhãn. Nguyên nhân do trong file annotation chỉ lưu chỉ số (0,1,3,4,...) của nhãn chứ không lưu tên nhãn.

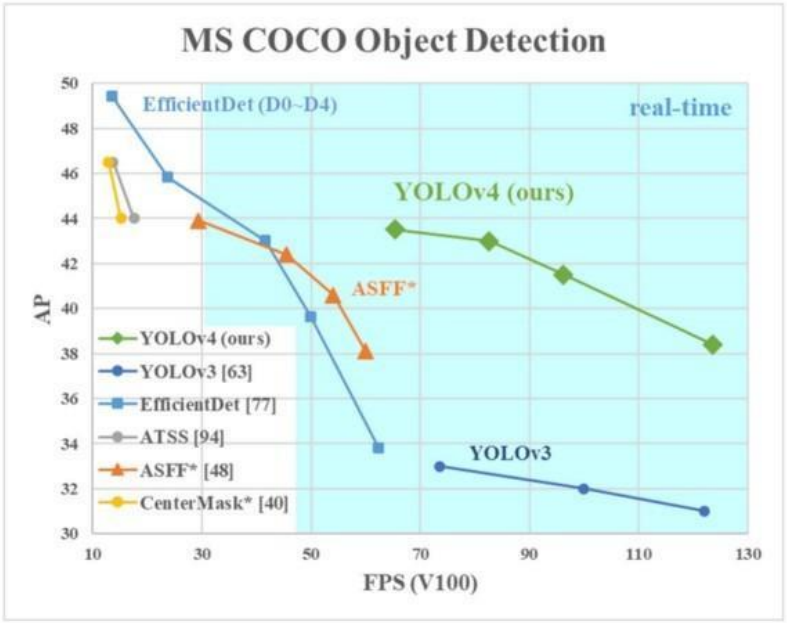
Sau khi gán nhãn xong các bạn để file annotation và ảnh tương ứng **vào cùng một thư mục**.

3. Huấn luyện mô hình

3.1 Giới thiệu về YOLO-Tinyv4 và darknet

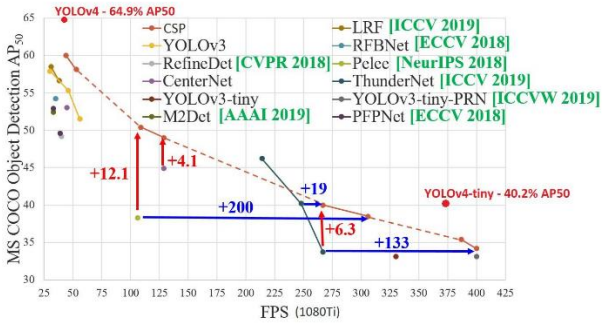
3.1.1 YOLO-Tinyv4

YOLOv4 là thuật toán Object Detection, mới được công bố trong thời gian gần đây với sự cải thiện về kết quả đáng kể so với YOLOv3.

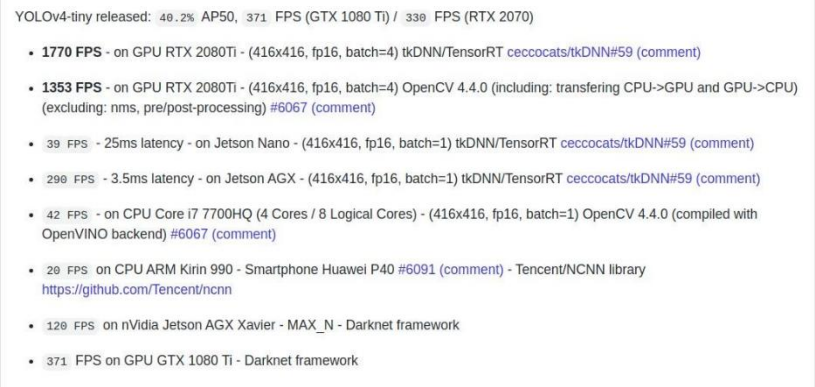


Hình 14.8: Sự cải thiện của YOLOv4 (source)

YOLOv4 cho kết quả real-time khi chạy trên các nền tảng GPU cao cấp. Với mục đích trade-off giữa độ chính xác và tốc độ để có thể chạy trên các nền tảng CPU và GPU thấp hơn thì YOLO-Tinyv4 được ra đời.



Hình 14.9: YOLOv4 với YOLO-Tinyv4 (source)



Hình 14.10: YOLO-Tinyv4 trên các nền tảng (source)

3.1.2 Darknet

Darknet là một framework open source chuyên về Object Detection được viết bằng ngôn ngữ C và CUDA. Darknet dùng để huấn luyện các mô hình YOLO một cách nhanh chóng, dễ sử dụng.