

TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN – ĐHQG TP HCM

-----oOo-----



UIT
TRƯỜNG ĐẠI HỌC
CÔNG NGHỆ THÔNG TIN

ĐỒ ÁN MÔN HỌC MÁY HỌC

**Đề tài: Nhận dạng các loại biển báo giao thông
Việt Nam bằng YOLOv4**

Giảng viên hướng dẫn: Lê Đình Duy
Phạm Nguyễn Trường An

Lớp: CS111.L22.KHCL

Nhóm thực hiện:

Từ Trương Tuấn Kiệt - 19521727

Võ Khoa Nam – 19521877

Phạm Trung Hiếu - 19521512

TP Hồ Chí Minh, ngày 30, tháng 06, năm 2021

MỤC LỤC

Chương 1. Tổng quan	3
1.1 Mô tả bài toán	3
1.1.1 Ngữ cảnh.....	3
1.1.2 Input và Output	3
1.2 Mô tả dữ liệu.....	4
Chương 2. Các nghiên cứu trước.	5
Chương 3. Xây dựng bộ dữ liệu.....	6
3.1 Tiêu chuẩn phân loại ảnh.....	6
3.2 Chuyển định dạng COCO format sang YOLO format	8
3.3 Kết quả.....	10
Chương 4. Training và đánh giá model	13
4.1 Mô hình mạng YOLOv4 và Backbone Darknet-53.....	13
4.2 Cài đặt và huấn luyện mô hình.....	15
4.2.1 Clone và thiết lập môi trường train	15
4.2.2 Chia tập train/valid và tạo các file yêu cầu	16
4.2.3 Huấn luyện model	18
4.2.4 Đánh giá model	19
Chương 5. Ứng dụng và hướng phát triển.	25

Chương 1. Tổng quan

1.1 Mô tả bài toán

1.1.1 Ngữ cảnh

Thế giới hiện nay đang phát triển với nhiều công nghệ hiện đại để giúp ích cho cuộc sống của con người. Trong đó xe tự hành là phát minh được thế giới quan tâm hiện nay, nó đem lại những thay đổi tích cực trong cuộc sống con người. Hiện nay, có rất nhiều hãng sản xuất xe hơi tự hành ví dụ như Tesla, Google,...Để có thể phát triển và vận hành được xe tự hành có rất nhiều vấn đề đặt ra. Trong số đó là vấn đề nhận diện biển báo giao thông. Với mong muốn ứng dụng kiến thức đã học trong môn học này vào thực tế. Nhóm chúng em đã chọn đề tài nhận diện biển báo giao thông Việt Nam cho phát triển xe tự hành.

1.1.2 Input và Output

Input: Video stream, hình ảnh trích xuất từ camera của xe ô tô có chứa biển báo giao thông.

Output: Video, hình ảnh với các bounding box của các loại biển báo giao thông Việt Nam.

Input:



Hình 1. Hai loại biển báo giao thông được cắt từ 1 frame hình

Output:



Hình 2. Hai loại biển báo giao thông được gắn nhãn với bounding box được cắt từ 1 frame hình

1.2 Mô tả dữ liệu

Việc tự mỗi thành viên trong nhóm thu thập dữ liệu gặp khá nhiều khó khăn khi dịch bệnh Covid đang hoành hành và lệnh giãn cách xã hội. Ngoài ra ba thành viên của nhóm ở xa nhau nên không có cơ hội gặp mặt trực tiếp trao đổi để có chung cách thức thu thập cũng như chất lượng mẫu ảnh thu thập được đồng nhất.

Vì thế nhóm quyết định sẽ tham khảo, sử dụng bộ dữ liệu biển báo giao thông Việt Nam của Zalo.AI với 5086 mẫu ảnh chụp ở nhiều khoảng cách dưới góc độ của 1 camera hành trình thích hợp cho mô hình bài toán, kích thước mẫu ảnh 1622 x 626



Chương 2. Các nghiên cứu trước.

Trước khi bắt đầu đề án này, nhóm chúng em đã tham khảo từ những người đã train về đề tài nhận diện biển báo giao thông như:

Nhóm của các anh Trương Quốc Bảo, Trương Hùng Chen và Trương Quốc Định của Trường Đại học Cần Thơ. Đề án này được các tiền bối thực hiện vào năm 2015 với bộ dữ liệu gồm 4 nhóm biển báo: biển báo cấm, biển báo nguy hiểm thuộc tập lệnh 1 và biển báo hiệu lệnh, biển báo chỉ dẫn thuộc tập lệnh 2 với tổng cộng 16257 mẫu ảnh tự thu thập. Kết quả sau khi test rất ấn tượng với tỉ lệ chính xác là 93.6% ở tập lệnh 1 và 99.41% ở tập lệnh 2 khi dùng phương pháp Multi-layer Perceptron (MLP) với đặc trưng HOG. Khi dùng mô hình SVM với đặc trưng HOG cũng đạt độ chính xác cao lần lượt 94.35%, 99.63% ở 2 tập lệnh.

Theo nhóm em nghiên cứu này có kết quả rất tốt khi toàn bộ dataset tập train, test đều được thu thập bằng máy ảnh có chất lượng cao, ánh sáng thích hợp, các biển báo được chụp chính diện không ở các góc khuất hay quá xa khiến việc học những đặc trưng dễ dàng hơn dẫn đến kết quả tốt.



Hình 3. Một số kết quả phát hiện và nhận dạng biển báo giao thông đường bộ

Chương 3. Xây dựng bộ dữ liệu

3.1 Tiêu chuẩn phân loại ảnh

Bộ dữ liệu nhóm chúng em sử dụng của Zalo.AI bao gồm 5086 mẫu ảnh với các nhãn đã được gán sẵn bởi định dạng COCO (Common Objects in Context) được chia làm 7 lớp đối tượng phổ biến: cấm đi ngược chiều, cấm đỗ xe, cấm quay đầu xe, giới hạn tốc độ, biển báo cấm khác, biển báo nguy hiểm, biển báo chỉ dẫn.

Nhóm tiến hành lọc lại bộ dữ liệu nhằm lấy những mẫu ảnh có đủ đặc trưng sau:

- + Ảnh không bị quá mờ, bể hình
- + Ảnh không chứa bất kỳ biển báo nào



Hình 4. Không chứa bất kỳ biển báo nào



Hình 5. Ảnh mẫu bị nhòe, bể hình

3.2 Chuyển định dạng COCO format sang YOLO format

Tiếp theo nhóm chúng em tiến hành chuyển các thông số của bounding-box trong định dạng COCO sang định dạng YOLO để tiến hành train dataset:

- Định dạng COCO sau key 'annotation' là một dictionary lưu các thuộc tính của bounding box chứa chứa các biển báo giao thông trong mẫu ảnh phân loại theo 7 lớp nêu trên.

Ví dụ:

```
{'segmentation': [], 'area': 342, 'iscrowd': 0, 'image_id': 3, 'bbox': [880, 333, 19, 18], 'category_id': 2, 'id': 0}
```

Với:

- + 'segmentation': chứa thông tin về tọa độ x, y cho các đa giác bao quanh đối tượng
- + 'area': diện tích của bounding box chứa đối tượng
- + 'iscrowd': cho biết đối tượng là riêng lẻ (0) hay là của 1 nhóm đối tượng (1)
- + 'image_id': id của mẫu ảnh (tên)
- + 'bbox': tọa độ góc trái trên của mẫu ảnh cùng độ rộng độ dài tương ứng [x, y, width, height]
- + 'category_id': lớp đối tượng được phân loại trong đó:
 - 0: Cấm đi ngược chiều
 - 1: Cấm đỗ xe
 - 2: Cấm quay đầu xe
 - 3: Giới hạn tốc độ
 - 4: Biển báo cấm khác
 - 5: Biển báo nguy hiểm
 - 6: Biển báo chỉ dẫn
- + 'id': id trong annotations

- Định dạng YOLO là file .txt tương ứng với mỗi mẫu ảnh trong đó gồm các thông số:

class_id	x	y	width	height
----------	---	---	-------	--------

- + class_id: lớp đối tượng được phân loại tương ứng với key 'category_id' trong định dạng COCO
- + x: tọa độ trung tâm của bounding box theo trục x / chiều rộng hình
- + y: tọa độ trung tâm của bounding box theo trục y / chiều cao hình
- + width: chiều rộng của bounding box / chiều rộng hình
- + height: chiều cao của bounding box / chiều cao hình

Hình 6. Định dạng YOLO Darknet của mẫu ảnh

Để thực hiện chuyển định dạng COCO sang định dạng YOLO nhóm chúng em dùng đoạn code sau:

```
def get_img_shape(path):
    img = cv2.imread(path)
    try:
        return img.shape
    except AttributeError:
        print("error! ", path)
        return (None, None, None)

def convert_labels(path, x1, y1, x2, y2):
    def sorting(l1, l2):
        if l1 > l2:
            lmax, lmin = l1, l2
            return lmax, lmin
        else:
            lmax, lmin = l2, l1
            return lmax, lmin
    size = get_img_shape(path)
    xmax, xmin = sorting(x1, x2)
    ymax, ymin = sorting(y1, y2)
    dw = 1.0/float(size[1])
    dh = 1.0/float(size[0])
    x = (xmin + xmax)/2.0
    y = (ymin + ymax)/2.0
    w = xmax - xmin
    h = ymax - ymin
    x = x*dw
    w = w*dw
    y = y*dh
    h = h*dh
    return (x,y,w,h)

#Tập check_set những image đã được lưu
check_set = set()
for i in range(len(training_data['annotations'])):
    image_id = str(training_data['annotations'][i]['image_id'])
    category_id = str(training_data['annotations'][i]['category_id'])
    number_of_sign[int(category_id)-1] += 1
    bbox = training_data['annotations'][i]['bbox']
    image_path = "images/" + image_id + ".png"
```

```

kitti_bbox = [bbox[0], bbox[1], bbox[2] + bbox[0], bbox[3] + bbox[
1]]
yolo_bbox = convert_labels(image_path, kitti_bbox[0], kitti_bbox[1
], kitti_bbox[2], kitti_bbox[3])
filename = "images/" + image_id + ".txt"
content = str(int(category_id) - 1) + " " + str(yolo_bbox[0]) + "
" + str(yolo_bbox[1]) + " " + str(yolo_bbox[2]) + " " + str(yolo_bbox
[3])

if image_id in check_set:
    # Thực hiện ghi đè lên file những chỉ số bounding box mới
    file = open(filename, "a")
    file.write("\n")
    file.write(content)
    file.close()
elif image_id not in check_set:
    check_set.add(image_id)
    # Tạo và ghi vào file
    file = open(filename, "w")
    file.write(content)
    file.close()

```

3.3 Kết quả

Dataset sau khi được phân loại chuyển định dạng vẫn giữ nguyên được độ đa dạng cũng như đáp ứng được các tiêu chuẩn đã nêu trên. Một vài ví dụ:



Hình 7. Mẫu ảnh chứa biển báo



Hình 8. Mẫu ảnh chứa biển báo ở khoảng cách lý tưởng để nhận dạng



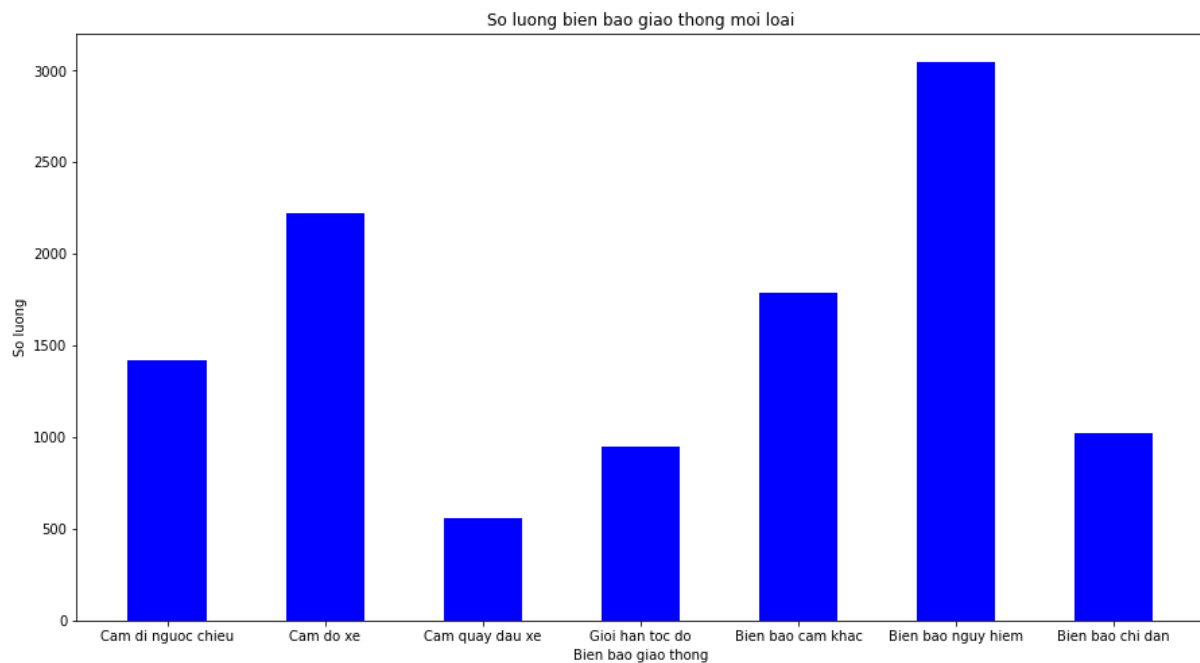
Hình 9. Mẫu ảnh bị khuất ngoài frame hình



Hình 10. Mẫu ảnh chứa 5 biển báo

Trong đó số lượng mỗi class trong 4500 mẫu ảnh được liệt kê như sau:

Class	Số lượng
Cấm đi ngược chiều	1416
Cấm đỗ xe	2221
Cấm quay đầu xe	556
Giới hạn tốc độ	949
Biển báo cấm khác	1787
Biển báo nguy hiểm	3049
Biển báo chỉ dẫn	1022



Hình 11. Đồ thị biểu diễn số lượng mẫu ảnh cho từng class

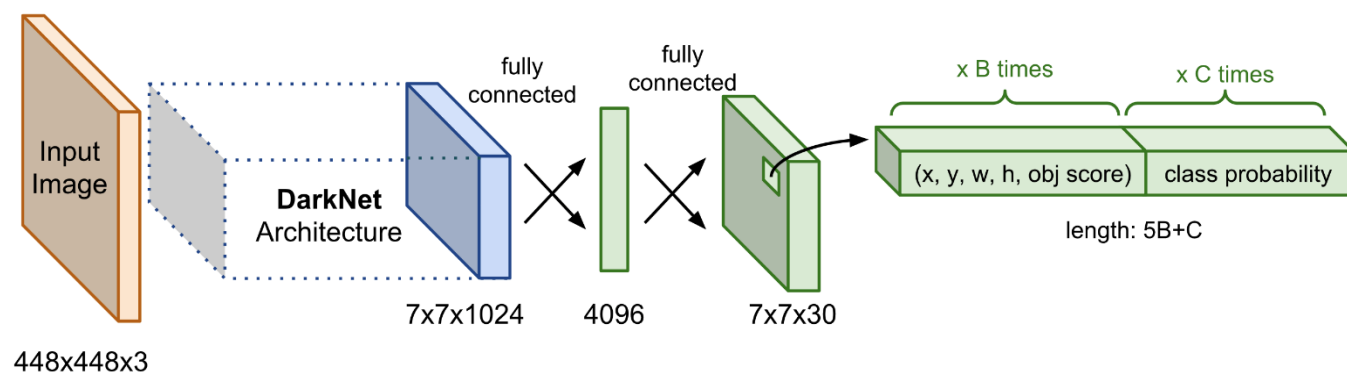
Chương 4. Training và đánh giá model

4.1 Mô hình mạng YOLOv4 và Backbone Darknet-53

YOLO (You Only Look Once) là một thuật toán dựa trên các lớp của mạng neuron tích chập để nhận dạng các vật thể. Cũng như cái tên của chính nó – chỉ cần nhìn một lần, thuật toán YOLO có thể nhận dạng đa dạng các vật thể thuộc nhiều lớp đối tượng.

Lý do nhóm chọn mô hình mạng YOLO vì tốc độ nhận dạng vật thể của nó. Tuy rằng độ chính xác có thể thua một số loại mạng CNN mới ra đời như FasterCNN, RCNN,... nhưng trong tình huống nhận dạng biển báo giao thông Việt Nam thì nhóm chúng em nghĩ vẫn nên ưu tiên tốc độ nhận dạng từ xa để tài xế có thể biết và tránh vi phạm luật giao thông.

Darknet-53 là một backbone được dùng phổ biến khi train mô hình YOLO. Darknet-53 gồm 53 convolutional layers kết nối liên tiếp, mỗi layer được theo sau bởi một batch normalization và một activation Leaky Relu



Hình 12. Sơ đồ kiến trúc mạng YOLO. Thành phần Darknet Architecture được gọi là base network có tác dụng trích xuất đặc trưng

*Một số khái niệm khác về model mạng YOLO

IoU (Intersection over union) là chỉ số đánh giá được sử dụng để đo độ chính xác của phát hiện đối tượng trên tập dữ liệu cụ thể. Chỉ số này thường được gặp trong các Object Detection Challenge. IOU thường được đánh giá hiệu năng của các bộ phát hiện đối tượng như HOG + Linear SVM và mạng nơ ron tích chập (R-CNN, FastR-CNN, YOLO,...).

Tỷ lệ IoU là tỉ lệ giữa đo lường mức độ giao nhau giữa hai đường bao (thường là đường bao dự đoán và đường bao thực) để nhằm xác định hai khung hình có bị đè chồng lên nhau không. Tỷ lệ này được tính dựa trên phần diện tích giao nhau giữa

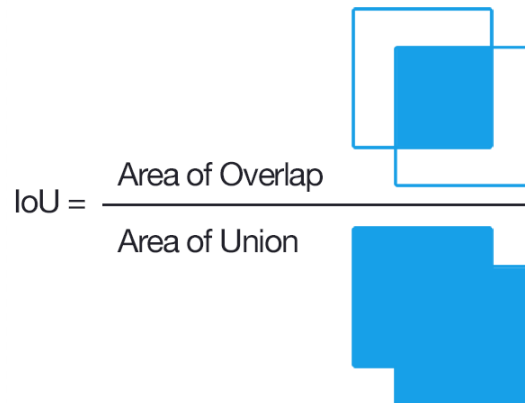
2 đường bao với phần tổng diện tích giao nhau và không giao nhau giữa chúng.

Gồm 3 giá trị:

+ $\text{IoU} > 0.5 \Rightarrow \text{True Positive (TP)}$

+ $\text{IoU} < 0.5 \Rightarrow \text{False Positive (FP)}$

+ $\text{IoU} = 0 \Rightarrow \text{False Negative (FN)}$ hay đối tượng không bị nhận dạng



Precision tỉ lệ số điểm Positive mô hình dự đoán đúng trên tổng số điểm mô hình dự đoán là Positive. Tỉ lệ này càng cao thì số điểm mô hình dự đoán là positive đều là positive càng nhiều. Nếu precision = 1, tức là tất cả số điểm mô hình dự đoán là Positive đều đúng, hay không có điểm nào có nhãn là Negative mà mô hình dự đoán nhầm là Positive.

Recall là tỉ lệ số điểm Positive mô hình dự đoán đúng trên tổng số điểm thật sự là Positive (hay tổng số điểm được gán nhãn là Positive ban đầu). Recall càng cao, tức là số điểm là positive bị bỏ sót càng ít. Nếu recall = 1, tức là tất cả số điểm có nhãn là Positive đều được mô hình nhận ra.

$$\text{Precision} = \frac{\text{True Positive}}{\text{True Positive} + \text{False Positive}}$$

$$\text{Recall} = \frac{\text{True Positive}}{\text{True Positive} + \text{False Negative}}$$

mAP hay viết tắt cho mean average precision là độ chính xác trung bình của model được tính bằng trung bình độ chính xác của tất cả các lớp.

4.2 Cài đặt và huấn luyện mô hình

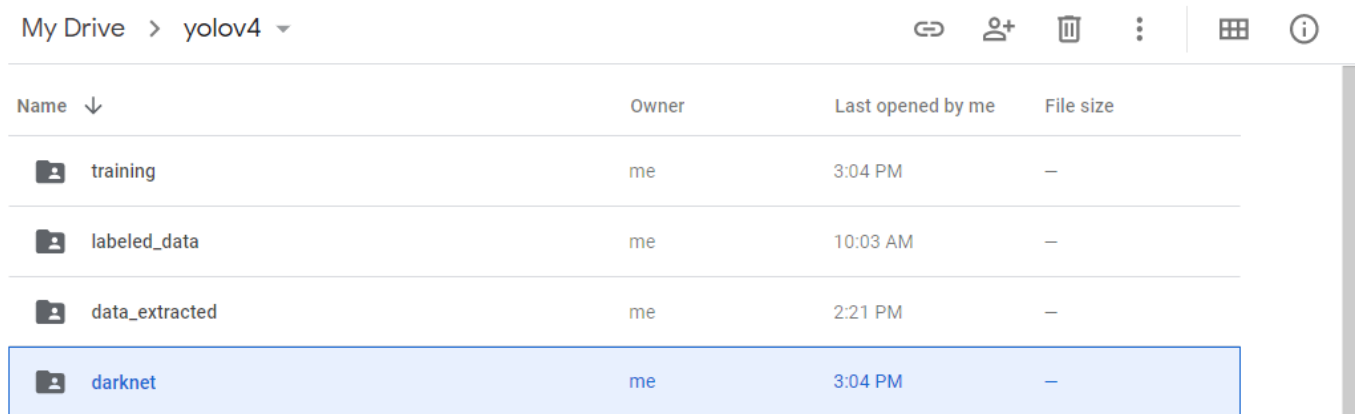
4.2.1 Clone và thiết lập môi trường train

Trước khi tiến hành nhóm chúng em đã tạo sẵn thư mục yolov4 và tải lên dataset gốc trong thư mục data_extracted và dataset đã gắn nhãn yolo format trong thư mục labeled_data có cấu trúc như sau:

```
yolov4/  
├─ labeled_data/  
└─ data_extracted/
```

Clone Darknet-53 từ github của tác giả AlexeyAB về Google Colab

```
!git clone https://github.com/AlexeyAB/darknet
```



The screenshot shows the Google Drive interface for a folder named 'yolov4'. The table lists the following files and folders:

Name	Owner	Last opened by me	File size
training	me	3:04 PM	—
labeled_data	me	10:03 AM	—
data_extracted	me	2:21 PM	—
darknet	me	3:04 PM	—

Hình 13. Thư mục darknet đã clone về drive

Nhóm chúng em tiến hành điều chỉnh 1 số thiết lập và cài đặt môi trường Darknet:

+ Thiết lập thông số cho GPU, CUDNN

```
!sed -i 's/OPENCV=0/OPENCV=1/' Makefile  
!sed -i 's/GPU=0/GPU=1/' Makefile  
!sed -i 's/CUDNN=0/CUDNN=1/' Makefile  
!sed -i 's/CUDNN_HALF=0/CUDNN_HALF=1/' Makefile  
!sed -i 's/LIBSO=0/LIBSO=1/' Makefile
```

+ Tạo môi trường darknet

```
!make
```

4.2.2 Chia tập train/valid và tạo các file yêu cầu

Sau khi môi trường darknet được khởi tạo nhóm chúng em tiến hành chia tập data thành 2 tập train / valid với tỉ lệ 9 / 1 với đoạn code:

```
import glob, os

curr_dir = os.path.dirname(os.path.abspath(__file__))

print(curr_dir)

curr_dir = 'data/labeled_data'

# Dataset có 4500 images nên sẽ chia train / valid = 9 / 1

# Tỉ lệ test
per_test = 10

# Tạo file .txt lưu địa chỉ image train valid
file_train = open('data/train.txt', 'w')
file_test = open('data/test.txt', 'w')

# Tiến hành chia sample theo mỗi 10 image thì image thứ 10 sẽ làm test
counter = 1
index_test = round(100 / per_test)
for path in glob.iglob(os.path.join(curr_dir, "*.png")):
    title, ext = os.path.splitext(os.path.basename(path))
    a = title + '.txt'
    b = "data/labeled_data" + "/" + a
    if counter == index_test and os.path.isfile(b):
        counter = 1
        file_test.write("data/labeled_data" + "/" + title + '.png' + "\n")
    elif counter != index_test and os.path.isfile(b):
        file_train.write("data/labeled_data" + "/" + title + '.png' + "\n")
    counter += 1
```

Lúc này cây thư mục trong folder darknet như sau:

```
yolov4/
├─ darknet/
│   └─ data/
│       ├── labels/
│       └─ train.txt
```



```
| | | test.txt
| | | split_data.py
| | | labeled_data/
```

Tiếp theo nhóm chúng em tiếp tục hoàn thành những file được yêu cầu để tiếp tục train: obj.data và obj.names.

+ File obj.data bao gồm các thông số của data như:

- classes: 7 #Số lượng class của model
- train = data/train.txt #Đường dẫn đến file train.txt
- valid = data/test.txt #Đường dẫn đến file test.txt
- names = data/obj.names #Đường dẫn đến file obj.names
- backup = /mydrive/yolov4/training .weights #Đường dẫn đến file .weights

+ File obj.names bao gồm tên hiển thị của data như:

- Cam di nguoc chieu
- Cam do xe
- Cam quay dau xe
- Gioi han toc do
- Bien bao cam khac
- Bien bao nguy hiem
- Bien bao chi dan

Tỉnh chỉnh file yolov4-custom.cfg. Đây là file config chính khi train model custom với các thông số được hiệu chỉnh cho phù hợp như sau:

+batch = 64 #Số lượng mẫu ảnh cho 1 lần train

+subdivisions = 16 #Số lượng mini-batches được chia nhỏ từ 64 batch đưa vào hay nói cách khác là $64/16 = 4$ mẫu ảnh cho 1 lần train

+width = 416 #Kích thước mẫu ảnh đưa vào được resize 416x416

+height = 416

+max_batches = 14000 #Số batch tối đa để train được tính bằng 2000 x số class

+steps = 11200, 12600 #Các step để model giảm learning_rate khi gần đạt đến điểm hội tụ = 80%, 90% max_batches

+learning_rate = 0.001 #Thông số học mặc định sẽ được hiệu chỉnh 2 lần để train

+filters = 36 #Chỉ chỉnh lại các filters ở cuối 3 base network theo công thức (classes + 5) x 3

4.2.3 Huấn luyện model

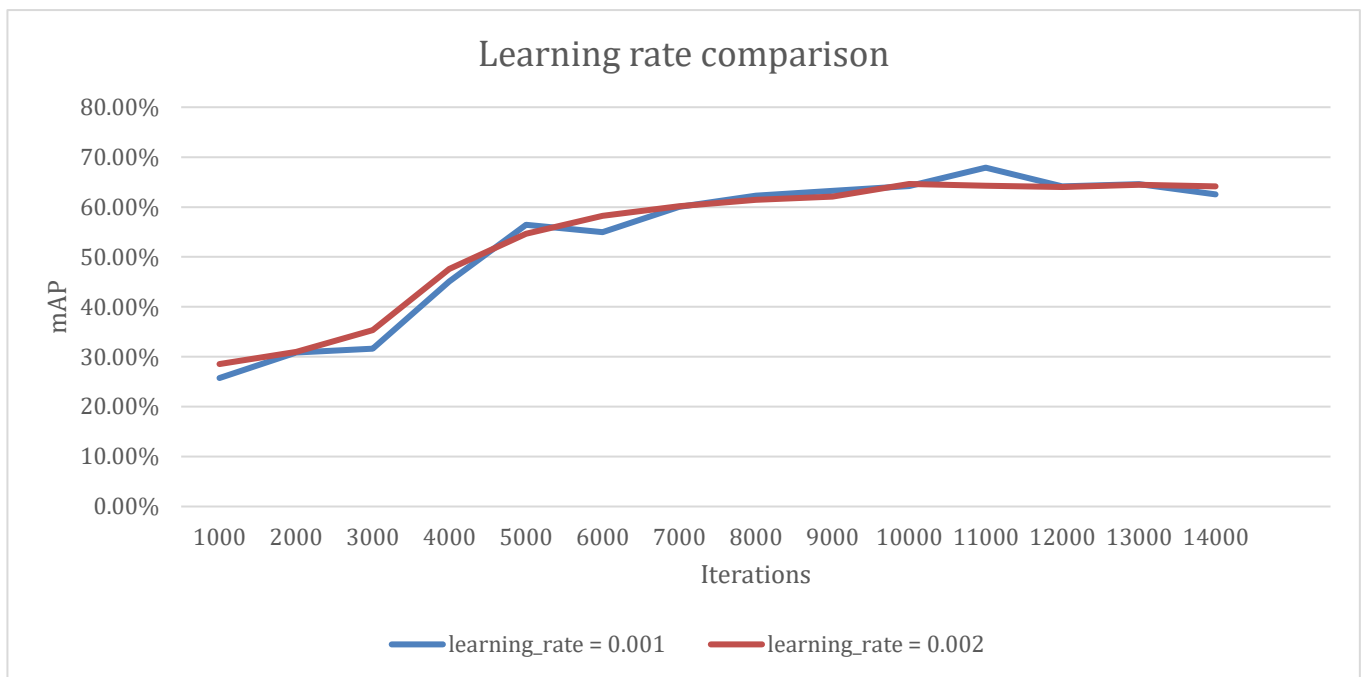
Để tiết kiệm thời gian cũng như tối ưu thuật toán nhóm sẽ sử dụng transfer learning với các feature đặc trưng đã được học trước. Những kiến thức đặc trưng đó được lưu trong file pre-trained nhóm sẽ sử dụng là yolov4.conv.137 với 137 lớp tích chập.

```
#Sử dụng pre-trained YOLOv4 weights 137 convolutional layer
!wget https://github.com/AlexeyAB/darknet/releases/download/darknet_yolo_v3_optimal/yolov4.conv.137
```

Sau khi hoàn tất cài đặt các packages, tạo môi trường darknet và cài đặt tinh chỉnh các file config thì tiếp theo nhóm tiến hành train với 2 learning_rate khác nhau: 0.001 và 0.002.

Vì Google Colab có giới hạn tài nguyên khi sử dụng GPU trong 7 tiếng liên tiếp và thời gian time out khá lâu (4-5 tiếng) dẫn đến biểu đồ thể hiện mAP và avg_loss của model bị làm mới mỗi lần train tiếp nên các kết quả thu được ở bảng dưới đây do các thành viên trong nhóm ghi chép lại mỗi 1000 iterations.

Iteration	mAP learning_rate = 0.001	mAP learning_rate = 0.002
1000	25.74%	28.54%
2000	30.86%	31.00%
3000	31.65%	35.34%
4000	45.11%	47.57%
5000	56.46%	54.66%
6000	54.98%	58.24%
7000	60.04%	60.15%
8000	62.3%	61.47%
9000	63.21%	62.08%
10000	64.19%	64.62%
11000	67.89%	64.29%
12000	64.13%	63.98%
13000	64.59%	64.49%
14000	62.52%	64.14%

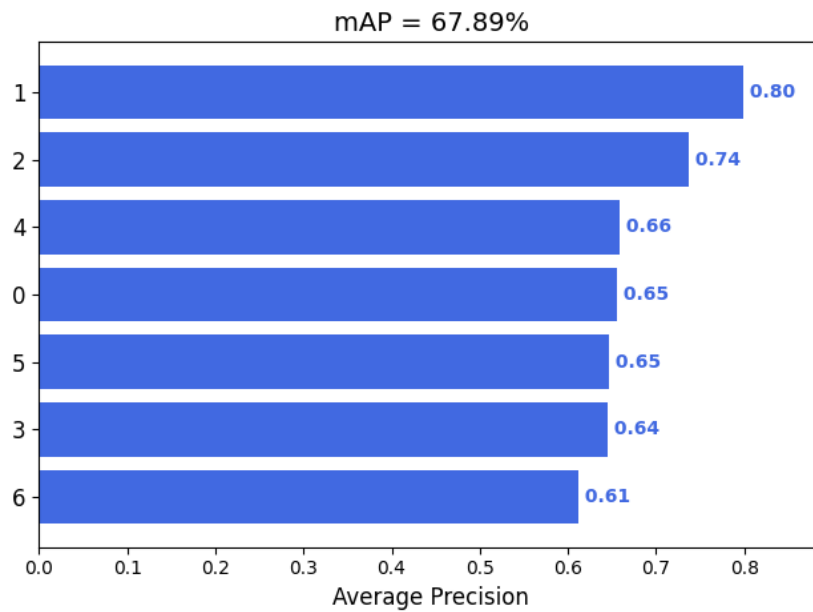


Hình 14. Đồ thị thể hiện mAP của model khi train bằng 2 learning rate khác nhau

4.2.4 Đánh giá model

Do một số sơ suất trong lúc training, các thành viên trong nhóm đã chỉ chú tâm vào con số mAP của mỗi 1000 iterations mà không ghi lại những tiêu chí khác như IoU, avg_loss. Do đó nhóm chỉ sẽ đánh giá model trên dựa trên tiêu chí mAP từ 2 lần train khác nhau.

Sau khi tăng learning_rate thì kết quả mAP thấp hơn so với để ở mặc định. Nhìn chung kết quả mAP của learning_rate = 0.001 vẫn còn quá thấp với mAP = 67.89%



Hình 15. Đồ thị biểu diễn AP từng class và mAP của model

*Nguyên nhân

Do dataset nhóm sử dụng của zalo.ai còn khá ít để chia đều cho 7 class dẫn đến những class có số lượng mẫu ít như cấm quay đầu xe hay giới hạn tốc độ bị nhận dạng sai khá nhiều cụ thể:



Hình 16. Hình gốc 1



Hình 17. Hình đã test 1

=>Hình gốc chứa 2 biển báo giới hạn tốc độ, 1 biển báo cấm đỗ xe và 1 biển báo cấm khác nhưng khi tiến hành nhận dạng thì biển báo cấm khác đã bị nhận dạng nhầm thành biển giới hạn tốc độ.

Nguyên nhân kế tiếp có thể là do chất lượng của mẫu hình chưa đến mức HD (1622x626) và các biển báo được gán bounding box đa số là ở xa so với khung hình nên chất lượng sau khi resize về kích thước 416x416 để đưa vào network train sẽ trở nên khó khăn hơn. Điều này có thể được kiểm chứng qua mAP của các iterations tăng không cao và không đáng kể dù đã sử dụng file pre-trained.



Hình 18. Hình gốc 2



Hình 19. Hình đã test 2

=>Các nhãn về giới hạn tốc độ bị hiểu sai và gán sai.

Tuy nhiên khi dùng model để nhận dạng các tấm ảnh có độ phân giải cao ổn định (3264x2448) với khoảng cách biển báo tương đương như mẫu ảnh ở Hình 18 thì kết quả khả quan hơn dù confidence (độ tự tin) khi nhận dạng biển báo giới hạn tốc độ còn thấp 0.57.

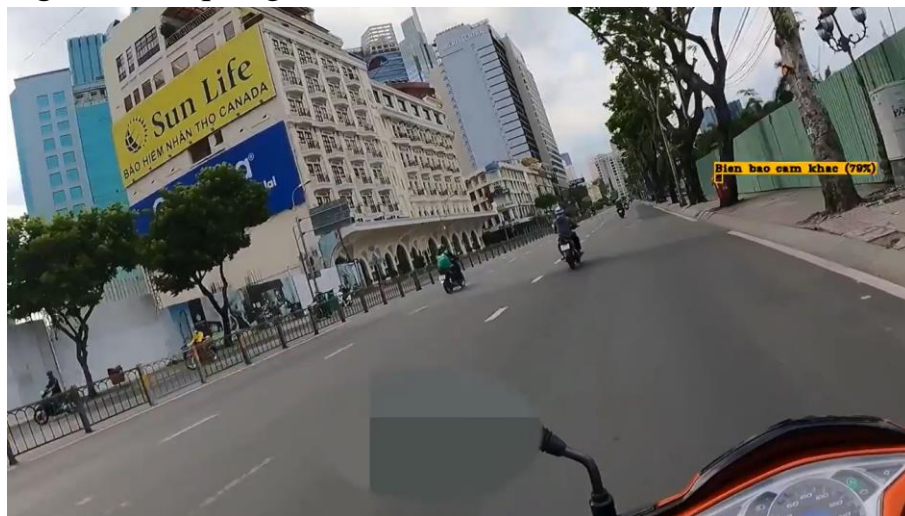


Hình 20. Hình gốc 3



Hình 21. Hình đã test 3

Ở mức độ nhận dạng trên video thì model nhận dạng chưa tốt dù có thể nhận dạng biển báo từ khoảng cách xa nhưng độ chính xác chưa cao và có sai sót. Model chỉ nhận dạng chính xác với confidence trên 0.9 với các biển báo cách camera hành trình 10m. Nhóm thực hiện nhận dạng trên 3 video được cắt từ youtube độ dài từ 40 giây đến 1 phút 30 giây độ phân giải Full HD, đạt 20 fps (frame per second). Kết quả nhận được tương đối khi demo 1 nhận dạng sai vào các vật thể giống biển báo như thùng rác biển quảng cáo.



Hình 22. Nhận dạng sai vật thể trích từ demo 1

Ở 2 demo còn lại nhận diện khá chắc chắn trong cự ly 10m.



Hình 23. Nhận dạng đúng với độ chính xác cao trích từ demo 2

Chi tiết demo video tại:

-Demo 1:

https://drive.google.com/file/d/1-pGa6dxuM-khVdlXo_b0r3KP9xqtWXcr/view?usp=sharing

-Demo 2:

<https://drive.google.com/file/d/1HIzOFgYO1G3RWx0PXDqlQF1yPAJBjt5J/view?usp=sharing>

-Demo 3:

https://drive.google.com/file/d/1ifHoPmz5pZYG_UO20R3iizfXtC16sbv/view?usp=sharing

Chương 5. Ứng dụng và hướng phát triển.

Về khả năng ứng dụng của model này áp dụng vào thực tế sẽ khó thành công bởi model:

- +Mất cân bằng về dữ liệu giữa các lớp
- +Chất lượng dataset để train chưa thật sự cao và phù hợp với thực tế
- +Các điều kiện ánh sáng, bóng râm, khoảng cách chưa đa dạng trong mẫu

ảnh train

Tuy nhiên nếu có đủ điều kiện để tận tay đi thu thập các mẫu ảnh (do nhóm chúng em đã thay đổi đề tài 2 lần thời gian để thực hiện đề tài này quá gần deadline và còn trong thời gian giãn cách) thì rất có thể kết quả sẽ khả quan hơn.

Về hướng phát triển sau khi model được hoàn tất với kết quả thực nghiệm cao thì sẽ có thể phát triển các thiết bị camera hành trình vừa nhận dạng vừa có thể phát ra tên biển báo ở khoảng cách trước mặt hạn chế được tình trạng tài xế không có khả năng nhìn các biển báo ở góc khuất, quá xa tầm mắt hay vào ban đêm. Tuy nhiên để thực hiện được điều này nhóm sẽ cần cải thiện model nhiều lần và học tập nhiều model khác, chuyên ngành khác trong Machine Learning.

TÀI LIỆU THAM KHẢO

[1] Dataset biển báo giao thông Việt Nam của zalo.ai,

https://miai.vn/download.php?url=http://www.mediafire.com/file/nbge6ccitmjd0th/zalo_tracffic_sign.zip/file

[2] Nghiên cứu của các sinh viên Đại học Cần Thơ năm 2005,

https://www.researchgate.net/publication/286927417_PHAT_HIEN_VA_NHAN_DANG_BIEN_BAO_GIAO_THONG_DUONG_BO_SU_DUNG_DAC_TRUNG_HOG_VA_MANG_NORON_NHAN_TAO

[3] Giới thiệu về mô hình YOLO của tác giả Phạm Đình Khánh năm 2020,

<https://phamdinhhkhanh.github.io/2020/03/09/DarknetAlgorithm.html>

[4] Tham khảo code chuyển định dạng coco sang định dạng yolo,

<https://medium.com/@thamqianyu96/coco-to-yolo-annotations-9d638bb3eb4f>

[5] Bài train mẫu YOLOv4 của Techzizou, [https://medium.com/analytics-](https://medium.com/analytics-vidhya/train-a-custom-yolov4-object-detector-using-google-colab-61a659d4868)

[vidhya/train-a-custom-yolov4-object-detector-using-google-colab-61a659d4868](https://medium.com/analytics-vidhya/train-a-custom-yolov4-object-detector-using-google-colab-61a659d4868)

[6] Bài train mẫu YOLOv4 của Jacob Solawetz,

<https://blog.roboflow.com/training-yolov4-on-a-custom-dataset/>