Đếm số lượng và phân loại phương tiện giao thông

1st Nguyễn Phan Quốc Thiện
Đại học Công nghệ Thông tin - đại học Quốc gia Thành phố Hồ Chí Minh
Thành phố Hồ Chí Minh, Việt Nam
email: 20520775@gm.uit.edu.vn

2nd Đỗ Đức Thịnh
Đại học Công nghệ Thông tin – đại học Quốc gia Thành phố Hồ Chí Minh
Thành phố Hồ Chí Minh, Việt Nam
email: 20520780@gm.uit.edu.vn

3rd Nguyễn Cao Quốc
Đại học Công nghệ Thông tin – đại học Quốc gia Thành phố Hồ Chí Minh
Thành phố Hồ Chí Minh, Việt Nam
email: 20520723@gm.uit.edu.vn

Tổng quan: ***Mục đích của nghiên cứu này nhằm kiểm tra số lượng phương tiện giao thông hằng ngày ở Việt Nam. Nghiên cứu được thực hiện dựa trên các phương pháp học máy và thống kê, sử dụng mô hình YOLOv5 cho việc phân loại phương tiện giao thông và thuật toán Deep SORT để đếm số lượng phương tiện sau khi đã phân loại chúng. Dựa vào các góc máy quay khác nhau trên từng tuyến đường riêng biệt, ta có những cách giải quyết khác nhau bằng việc tinh chỉnh tham số của mô hình và thuật toán. Từ đó có thể giúp các doanh nghiệp, cửa hàng có các quyết định quảng cáo, đặt cửa hàng tại vị trí thuận lợi nhằm tăng doanh thu cũng như lợi nhuận. Từ kết quả phân tích thu được, ta thấy rằng những tuyến đường có mật độ phương tiện giao thông qua lại lớn thường là các tuyến đường lớn nằm trong nội thành của những thành phố phát triển, nằm gần các trung tâm thương mại lớn hoặc các khu dân cư đông đúc. Nhóm em đi đến kết luận rằng với mục đích tăng doanh số sản phẩm cũng như giúp lợi nhuận thu về nhiều hơn, các doanh nghiệp hay cửa hàng cần phải đặt chi nhánh hoặc cửa hàng tại các tuyến đường có mật độ lưu thông cao, có nhiều người qua lại trong ngày.***

Từ khóa – YOLOv5, Deep SORT, Vehicle Detection, Classification.

I. GIỚI THIỆU

Tại một đất nước đang phát triển như Việt Nam, mật độ lưu thông trên các tuyến đường của các phương tiện tham gia giao thông đang ngày một tăng lên do nhiều yếu tố khác nhau như kinh tế, xã hội hoặc văn hóa. Nhờ vậy mà việc xác định mật độ lưu thông cao hay thấp có thể mang lại nhiều lợi ích cho các doanh nghiệp kinh doanh hoặc cửa hàng buôn bán như là giúp đặt các biển quảng cáo và chi nhánh sao cho mang lại doanh thu hiệu quả. Mục tiêu cụ thể của nghiên cứu này nhằm mang tới một giải pháp ứng dụng cho người dùng có thể xác định xem mật độ lưu thông trên một tuyến đường nhất định là bao nhiêu, và có bao nhiêu số lượng mỗi loại xe lưu thông trong một khoảng thời gian xác định.

Để có thể xác định mật độ lưu thông, con người không thể thực hiện chỉ với đôi mắt vì mắt chúng ta không cho phép nhìn nhiều tuyến đường cùng một lúc và tập trung một cách hiệu quả. Bên cạnh đó thì có những loại phương tiện giao thông khác nhau như xe máy, xe đạp, xe hơi, xe buýt, xe khách, xe tải,…việc có nhiều loại phương tiện giao thông như thế trực tiếp khiến cho con người khó khăn trong việc phân loại và nhận diện tất cả cùng một lúc. Do đó yêu cầu chúng ta phải có một ứng dụng hoặc một hệ thống thông minh dễ sử dụng để làm việc đó thay cho con người chúng ta. Ứng dụng này trước tiên sẽ nhận diện xem một vật thể xuất hiện trong video có phải là một phương tiện tham gia giao thông hay không, sau đấy sẽ được phân loại thành các lớp tương ứng như là xe máy, xe hơi, xe buýt, xe tải,… bằng YOLO [1]. Nếu phương tiện ấy được nhận diện thì sẽ được theo dõi và đếm vào số lượng phương tiện tham gia giao thông bằng thuật toán Deep SORT [2]. Tốc độ nhận diện và độ chính xác của mô hình sẽ phụ thuộc một phần vào phiên bản của YOLO mà ứng dụng sử dụng. Những phiên bản YOLO mới nhất sẽ cân bằng giữa tốc độ nhận diện và độ chính xác tốt hơn. Nhận diện phương tiện giao thông là một bài toán trong lĩnh vực học máy, nơi mà các mô hình máy học có thể tự học và cải thiện việc nhận dạng, phân loại và đếm số lượng phương tiện giao thông. Trong nghiên cứu này, nhóm sẽ sử dụng ngôn ngữ Python, mô hình YOLO và thuật toán Deep SORT để giải quyết. Lí do sử dụng ngôn ngữ Python là vì Python có cú pháp ngắn gọn, dễ hiểu, đồng thời có nhiều thư viện cũng như framework hỗ trợ cho máy học.

II. DỮ LIỆU

A. THU THẬP HÌNH ẢNH

Vì đây là lĩnh vực máy học nên ta cần dữ liệu đầu vào để giúp huấn luyện mô hình. Trong nghiên cứu này chúng ta thu thập những tấm hình có nội dung chứa đựng phương tiện giao thông để sử dụng cho việc huấn luyện mô hình YOLO. Số lượng phương tiện giao thông trong từng tấm hình, sự đa dạng của các phương tiện giao thông cũng như số lượng phương tiện tham gia giao thông ảnh hưởng rất lớn tới độ chính xác của thuật toán. Toàn bộ dữ liệu bao gồm 764 ảnh. Tập dữ liệu huấn luyện có cấu trúc bao gồm 612 ảnh, trong đó bao gồm 11% là tự gán nhãn, 80% được lấy từ Github [3], 9% được lấy từ RoboFlow [4]. Dữ liệu từ Github và RoboFlow đã được gán nhãn sẵn với 4 nhãn chính là motorcycle, car, truck và bus. Trong đó có 1731 nhãn là car, 243 nhãn là motocycle, 250 nhãn là truck và 151 nhãn là bus.

B. DỮ LIỆU CHƯA CÓ NHÃN

Đây là tập dữ liệu nhóm thu thập được từ cuộc thi ICPC 2019 [5] và từ trang web UIT Together [6], dữ liệu bao gồm 82 tấm ảnh được cắt ra từ các video khác nhau trong các khoảng thời gian khác nhau. Đối với dữ liệu từ ICPC 2019, video được lấy từ camera đặt tại các tuyến đường lớn, đông người qua lại ở Việt Nam nên nhóm tự cắt ảnh từ những video ấy đồng thời bổ sung thêm vài tấm ảnh tự cắt từ trên youtube. Còn với dữ liệu từ trang web UIT Together, video có bối cảnh ở trên một đường cao tốc tại Trung Quốc có nhiều loại phương tiện tham gia giao thông.

C. DỮ LIỆU ĐÃ CÓ NHÃN

Đây là tập dữ liệu đã được gán nhãn sẵn mà nhóm thu thập từ trên hai nền tảng là Github và RoboFlow.

- Với tập dữ liệu từ Github, nhóm thu được 606 bức ảnh [7]. Là tập dữ liệu có số lượng lớn nhất trong 3 tập dữ liệu nhóm thu thập được.
- Với tập dữ liệu từ RoboFlow, nhóm thu được 76 bức ảnh [8]. Là tập dữ liệu có số lượng nhỏ nhất trong 3 tập dữ liệu nhóm thu thập được.

D. GÁN NHÃN DỮ LIỆU

Đối với tập dữ liệu chưa có nhãn, nhóm đã sử dụng website RoboFlow, là một website hỗ trợ gán nhãn các tấm hình khác nhau, để gán nhãn các bức ảnh chưa có nhãn ở mục *B*. Đầu tiên nhóm chọn lọc các bức ảnh có vật thể rõ ràng, ít hoặc không bị các vật thể khác che mất và các bức ảnh ít bị nhiễu. Kế đến nhóm đưa toàn bộ các bức ảnh đã lọc vào RoboFlow và thực hiện gán nhãn bằng tay do tập dữ liệu này không quá lớn, thực hiện bằng tay sẽ cho độ chính xác cao hơn tuy nhiên lại mất nhiều thời gian hơn do số lượng ảnh rất nhiều. Với mỗi vật thể được chọn, nhóm sẽ gán cho vật thể

ấy một nhãn tương ứng với 1 trong 4 nhãn là car, motorcycle, truck hoặc bus. Sau khi việc gán nhãn hoàn thành thì nhóm thực hiện việc đặt thứ tự cho các nhãn ở trong khâu Preprocessing: Modify Classes (đây là một tính năng của RoboFlow, giúp ánh xạ các Class thành dạng số tự nhiên để dễ dàng xử lý). Các nhãn được đặt cùng thứ tự với bộ dữ liệu đã được gán nhãn sẵn ở mục *C*, nhãn 0 tương ứng với car, nhãn 1 sẽ là motorcycle, nhãn 2 là truck và nhãn 3 là bus.
Cuối cùng nhóm trích xuất bộ dữ liệu mới ra một file riêng và kết hợp nó với bộ dữ liệu ở mục *C*.

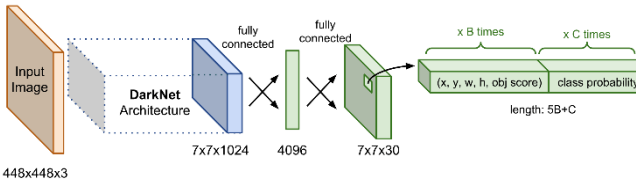
III. PHƯƠNG PHÁP MÁY HỌC

Mục này sẽ mô tả những phương pháp đã được sử dụng trong nghiên cứu này. Cụ thể là mô hình YOLOv5 và thuật toán Deep SORT.

A. YOLO LÀ GÌ ?

YOLO là viết tắt của cụm từ "You Only Look One", tức là mô hình chỉ cần nhìn một lần để phát hiện ra vật thể. YOLO là một mô hình mạng CNN sử dụng cho việc phát hiện, nhận dạng cũng như phân loại đối tượng. Cấu trúc của YOLO bao gồm base network là những mạng tích chập làm nhiệm vụ trích xuất đặc trưng. Phần phía sau là những Layer đặc biệt được áp dụng để phát hiện vật thể trên features map của base network [9].

Base network của YOLO sử dụng chủ yếu là các convolutional layer và các fully conntected layer. Các kiến trúc YOLO cũng khá đa dạng và có thể tùy biến thành các version cho nhiều input shape khác nhau.



Ảnh 1: Sơ đồ cấu trúc mạng YOLO

Hình trên là sơ đồ cấu trúc mạng YOLO. Thành phần Darknet Architecture được gọi là base network có tác dụng trích suất đặc trưng. Output của base network là một feature map có kích thước 7x7x1024 sẽ được sử dụng làm input cho các Extra layers có tác dụng dự đoán nhãn và tọa độ bounding box của vật thể.

B. CÁCH MÔ HÌNH YOLO HOẠT ĐỘNG

Mô hình YOLO hoạt động theo trình tự 3 bước như sau [10]:

- Xác định khối nhận diện

Đầu tiên, hình ảnh được chia thành nhiều ô khác nhau. Mỗi ô có số chiều là S x S. Hình ảnh dưới đây cho ta thấy rằng hình ảnh đầu vào được chia thành các ô ra sao.

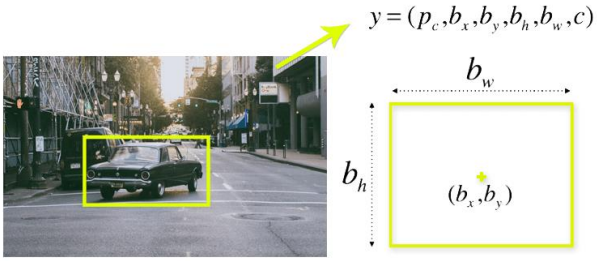
Ảnh 2: Cách mô hình chia ô

Trong hình ảnh trên, các ô được chia ra có chiều dài bằng chiều cao. Mỗi ô sẽ nhận diện một đối tượng xuất hiện ở trong ô. Ví dụ, nếu điểm trung tâm của đối tượng xuất hiện ở trong một ô nhất định thì ô này sẽ nhận trách nhiệm nhận diện đối tượng đó.

- Tìm bounding box

Mỗi ô có đầu ra gồm 6 thông tin: xác suất tồn tại đối tượng trong ô (pc), thông tin của bounding box và xác suất đối tượng thuộc một nhãn (c).

Một bounding box là một viền kẻ bên ngoài của một đối tượng trong tâm ảnh. Mỗi bounding box đều chứa những thông tin sau đây: chiều rộng (bw), chiều cao (bh), tâm của bounding box (bx, by). Hình ảnh dưới đây là ví dụ của 1 bounding box, bounding box được diễn tả bằng đường kẻ vàng.



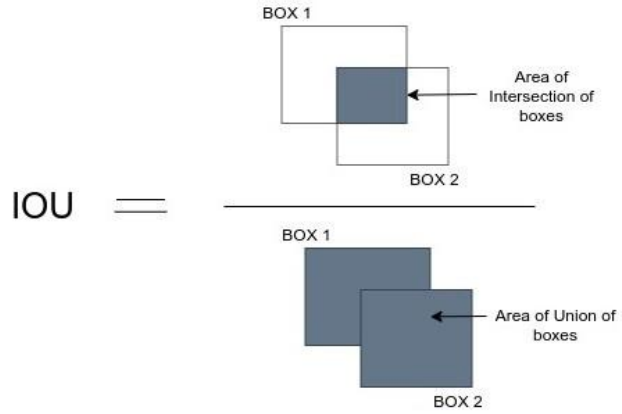
Ảnh 3: Ví dụ của bounding box

- Độ đo đánh giá mô hình (IOU)

IOU là độ đo đánh giá các mô hình nhận diện đối tượng. Phép đo này có thể đánh giá các mô hình khác nhau như RCNN, Faster-RCNN hay YOLO [11].

IOU là một thông số được sử dụng để đánh giá độ che lấp lên nhau giữa hai bounding box.[12].

IOU được tính theo công thức sau:



Ảnh 4: Công thức độ đo IOU

- Non-Maximum Suppression

Thông thường output của các mô hình Object Detection sẽ có rất nhiều các bounding box, việc đó sẽ gây ra hiện tượng có nhiều bounding box cho cùng một đối tượng, từ đó gây ra hiện tượng dư thừa thông tin khi mà mục đích của ta chỉ cần duy nhất một bounding box cho mỗi đối tượng. Vậy nên ta sẽ sử dụng thuật toán Non-Maximum Suppression để loại bỏ đi các bounding box dư thừa và chỉ giữ lại một bounding box cho mỗi đối tượng. [13]

Nội dụng thuật toán:

- Input: một mảng các bounding box, mỗi bounding box có dạng (x1, y1, x2, y2, c) trong đó:
  - (x1, y1) và (x2, y2) lần lượt là tọa độ điểm top-left và bottom-right của bounding box.
  - c là confidence score tương ứng với box đó, được trả về từ mô hình object detection.
- Output: Một mảng các bounding box sau khi đã được loại bỏ đi các bounding box dư thừa.

Chi tiết thuật toán:

- S: bounding box đang xét
- P: tập các box đầu vào của thuật toán
- Thresh_iou: ngưỡng IOU để loại bỏ các bounding box thừa
- Keep: tập các bounding box sau khi đã loại bỏ các bounding box thừa

Thuật toán bao gồm 3 bước:

1. Chọn bounding box S có confidence score cao nhất trong tập P, loại bỏ box đó ra khỏi tập P và thêm box đó vào tập Keep.
2. Thực hiện tính toán IOU giữa box S vừa lấy ra ở bước 1 với toàn bộ các box còn lại trong tập P. Nếu có box nào trong P có IOU với box S đang xét mà lớn hơn ngưỡng Thresh_iou thì loại bỏ box đó ra khỏi P.
3. Lặp lại bước 1 cho đến khi P không còn box nào.

Sau khi kết thúc thì tập Keep giữ lại toàn bộ các box sau khi đã loại bỏ các box thừa.

Identify applicable funding agency here. If none, delete this text box.

C. TẠI SAO LẠI CHỌN MÔ HÌNH YOLOV5?

Thứ nhất, YOLOv5 là phiên bản YOLO được tạo ra bằng Pytorch thay vì tạo ra bằng Darknet của PJ Reddie. Darknet là một framework nghiên cứu vô cùng linh hoạt nhưng nó không được xây dựng để phù hợp với các doanh nghiệp làm sản phẩm, bên cạnh đó nó có một cộng đồng người sử dụng vô cùng ít ỏi. Từ những điều kể trên dẫn đến việc Darknet có nhiều khó khăn hơn trong việc sử dụng và có ít tác dụng hơn cho các doanh nghiệp làm sản phẩm [14].

Do YOLOv5 được triển khai bằng Pytorch nên nó được Pytorch hỗ trợ nhiều hơn: hỗ trợ đơn giản hơn, triển khai dễ dàng hơn. Hơn nữa với việc là một framework được biết đến rộng rãi hơn, Pytorch có thể giúp cộng đồng nghiên cứu dễ dàng tiếp cận với YOLOv5 hơn. Điều này cũng giúp cho việc triển khai ứng dụng có YOLOv5 trên các thiết bị di động trở nên đơn giản hơn do mô hình có thể được biên dịch thành dạng ONNX và CoreML một cách dễ dàng.

Thứ hai, YOLOv5 nhẹ hơn các phiên bản YOLO khác. YOLOv5 chỉ nặng khoảng 27 megabytes. Trong khi YOLOv4 được xây dựng bằng Darknet có khối lượng lên tới 244 megabytes. YOLOv5 nhỏ hơn gần 90% so với YOLOv4. Điều này cho phép YOLOv5 nhúng vào các loại thiết bị khác nhau dễ dàng hơn rất nhiều [15].

D. THEO DÕI ĐA ĐỐI TƯỢNG

Theo Dadhich [16], theo dõi đa đối tượng (Multiple Objects Tracking – MOT) có thể chia thành hai bước: nhận diện và kết hợp. Một trong những thuật toán theo dõi đa đối tượng thường được sử dụng là Deep Simple Online and Real-time Tracking.

E. ĐỊNH NGHĨA DEEP SORT

Theo Dadhich [17], ta có thể định nghĩa Deep SORT theo ba bước sau:

a) Phát hiện đối tượng dựa trên CNN (trong nghiên cứu này là YOLO) dùng để nhận diện đối tượng trong khung hình.
b) Liên kết dữ liệu của một mô hình ước tính. Thuật toán này sử dụng một vector chứa tám tham số bao gồm tọa độ điểm trung tâm (x, y), chiều rộng (l), chiều cao (h) và đạo hàm của nó (x', y', l', h') để thực hiện theo dõi đối tượng. Bộ lọc Kalman được sử dụng để mô hình hóa các thuật toán này như là một hệ thống động.
c) Từ dự đoán của bộ lọc Kalman, chúng ta liên kết đối tượng được nhận diện ở khung hình mới với đối tượng được nhận diện ở khung hình cũ. Sự liên kết này được tính toán bằng thuật toán Hungary (Hungarian Algorithm) với ma trận liên kết đo đạc các bounding box chồng lên nhau sau "n" khung hình.

IV. CÁC THỬ NGHIỆM TINH CHỈNH MÔ HÌNH

Việc thử nghiệm, tinh chỉnh các tham số của mô hình trong quá trình huấn luyện giúp cho nhóm có thể tìm ra cách tối ưu nhất cho mô hình huấn luyện, từ đó giúp nâng cao hiệu suất, kết quả của mô hình cho từng loại dữ liệu khác nhau.

A. HUẤN LUYỆN MÔ HÌNH

Mô hình mà nhóm chọn chính xác là YOLOv5n, kiến trúc đơn giản và kích thước nhỏ nhất, việc lựa chọn mô hình nhỏ

sẽ có tốc độ nhanh hơn nhưng phải đánh đổi một ít ở độ chính xác. Bởi vì nhóm không có GPU mạnh cho việc huấn luyện và kiểm thử mô hình cũng như nhằm mục đích thuận tiện, rút ngắn thời gian cho việc huấn luyện mô hình, nhóm đã sử dụng Google Colab bản miễn phí để sử dụng GPU NVIDIA K80 của Google. Các thông số khi huấn luyện mô hình là 200 epochs với batch size là 64, kích cỡ của ảnh là 640x640 đã cho ra kết quả sau khi đánh giá trên tập test gồm 76 ảnh như hình bảng dưới đây:

CLASS	PRECISION	RECALL
ALL	0.918	0.927
CAR	0.942	0.954
MOTORCYCLE	0.821	0.909
TRUCK	0.91	0.864
BUS	1	0.982

Bảng 1: Độ chính xác cho từng lớp

B. KẾT HỢP VỚI DEEP SORT

Với mỗi xe được YOLO nhận diện và phân loại thì sẽ được Deep SORT gán cho 1 ID để theo dõi. Nhóm sẽ đếm bằng cách tạo ra 1 vạch kẻ trong khung hình, khi xe đã đi qua vạch kẻ đó thì sẽ thêm ID của nó vào 1 trong 4 mảng car, motorcycle, truck, bus. Số lượng ID của mỗi mảng cũng là số lượng của từng loại xe.

C. HIỆU SUẤT MÔ HÌNH

Với mong muốn phát triển xa hơn thay vì chỉ dừng lại ở mức mô hình, nhóm đã xây dựng thành demo nhỏ với sự giúp đỡ của framework Streamlit [18]. Phần cứng dùng để chạy mô hình là CPU Intel I5-8350U, GPU tích hợp Intel UHD 620, 8GB RAM với BUS 2400MHz, ổ cứng SSD 256GB. Sau khi đã chạy thử nghiệm trên 3 video có cùng độ dài về thời lượng là 10 giây, trong đó có 1 video có bối cảnh là một đoạn đường với ít xe lưu thông (traffic10s.mp4), 1 video có bối cảnh là một đường cao tốc với mật độ phương tiện giao thông khá lớn (test10s.mp4) và 1 video có bối cảnh đặt tại một cây cầu nhỏ có vị trí tại Việt Nam với mật độ phương tiện tham gia giao thông cao, nhưng chủ yếu là xe gắn máy. Cả 3 video trên đều được trích xuất từ các camera theo dõi tại các tuyến đường nơi nó ghi hình. Kết quả hiệu suất của mô hình cho ra như bảng dưới đây:

VIDEO	FPS TRUNG BÌNH	PROCESSING SPEED (MS)			
		PRE-PROCESS	INFERRENCE	NMS	DEEP SORT UPDATE
Traffic 10s.mp 4	5.9	1.7	114.1	0.8	51.9
Test10 s.mp4	5.3	1.2	98	0.9	87.2
Vn10s. mp4	3.0	1.7	147.9	1.4	186

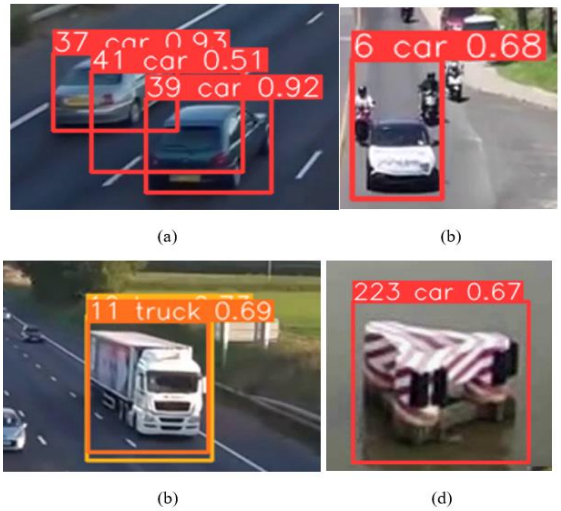
Bảng 2: FPS và tốc độ xử lý của mô hình

Từ kết quả trên, ta có thể thấy rằng thời gian cho hoạt động tiền xử lí (preprocess) và Non-Maximum Suppression (NMS) không quá đáng kể nhưng trái lại, thời gian suy luận của mạng YOLO (inference) và cập nhật lại Deep SORT (Deep SORT update) chiếm phần lớn thời gian xử lí. Theo suy đoán của nhóm, lí do cho việc xử lí lâu như vậy có thể là do mật độ xe lưu thông không đồng đều, càng nhiều xe thì mô hình càng phải dự đoán nhiều bounding box và thuật toán Deep SORT sẽ phải theo dấu nhiều xe hơn.

Do không có GPU rời nên tốc độ xử lý chậm dẫn đến việc FPS trung bình khá thấp, khó có thể chạy được trong thời gian thực, nơi cần tốc độ xử lí cực cao. Nếu muốn cải thiện về mặt FPS thì theo nhóm, cách tối ưu nhất là dùng GPU có hiệu năng cao.

V. PHÂN TÍCH LỖI, HƯỚNG PHÁT TRIỂN

A. PHÂN TÍCH LỖI



Ảnh 5: Một số trường hợp nhận dạng lỗi

Các trường hợp mà mô hình nhận dạng sai có thể phụ thuộc vào nhiều yếu tố:

(a) Chỉ có 2 car nhưng lại nhận dạng là 3 car, ID 41 bị nhận dạng sai chỉ có độ tin cậy là 0.51.

(b) Nhận dạng car và 2 motorcycle là 1 car. Trường hợp này xảy ra khi đối tượng nằm ở xa camera.

(c) Phân loại 1 đối tượng đều thuộc 2 phân lớp. Trường hợp này cũng xảy ra khi xe ở xa camera.

(d) Đối tượng không thuộc 1 trong 4 phân lớp nhưng vẫn được nhận dạng.

Trường hợp (a) có thể khắc phục bằng cách tăng siêu tham số độ tin cậy. Đối với trường hợp (b) và (c) xảy ra khi đối tượng ở xa camera, muốn đếm chính xác thì có thể điều chỉnh tham số là vị trí của vạch kẻ thấp xuống để nhận diện tốt hơn khi xe đi đến gần camera. Trường hợp (d) theo nhóm thì nên thu thập thêm dữ liệu để huấn luyện.

B. HƯỚNG PHÁT TRIỂN

a) Cải thiện mô hình và thêm tính năng

Để mô hình hoạt động tốt hơn và có thể cho ra kết quả tốt hơn trong thực tế thì nhóm sẽ tiếp tục nghiên cứu sâu hơn để tối ưu hóa mô hình về mặt tốc độ xử lí, độ chính xác của mô hình. Thu thập thêm các dữ liệu mới để dữ liệu đa dạng hơn đồng thời phát triển ứng dụng sao cho có thể đếm được xe ở riêng hai chiều ngược nhau.

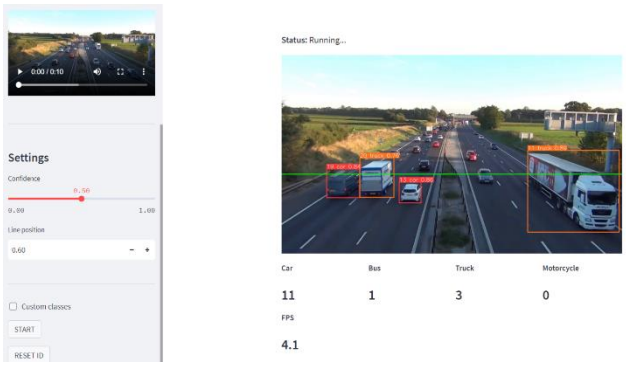
Cuối cùng là có thể vẽ được mô hình hiển thị mật độ xe tại từng đoạn đường khác nhau trong cùng thành phố, mục đích là để cảnh báo người dân nơi nào đang bị kẹt xe để có thể chọn những con đường khác để di chuyển.

b) Phát triển thành ứng dụng

Với mong muốn mang nghiên cứu có thể tới với mọi người, giúp mọi người kể cả là người không chuyên thì nhóm

đã phát triển nghiên cứu thành một web trên máy tính hoặc laptop bằng công cụ Streamlit, tuy nhiên hiện tại nhóm chỉ mới triển khai Streamlit trên local chứ chưa thể triển khai thành web server.

Streamlit là một framework được xây dựng với mục đích dành cho những kĩ sư trí tuệ nhân tạo có thể tạo ra một giao diện web như Jupyter Notebook. Điểm khác biệt với Jupyter Notebook chính là Streamlit không hiển thị code, thay vào đó Streamlit hiển thị một giao diện web trông giống như một sản phẩm có tính hoàn thiện cao.



Ảnh 6: Giao diện của web

VI. KẾT LUẬN

Bài báo cáo hướng đến việc xây dựng một mô hình máy học giúp nhận diện và phân loại các loại phương tiện tham gia giao thông tại Việt Nam dựa vào mô hình YOLOv5 và thuật toán Deep SORT. Quá trình nghiên cứu và thực nghiệm với video thực tế tại các thành phố lớn ở Việt Nam như Hồ Chí Minh và Hà Nội, nhóm nhận thấy khi mật độ lưu thông thấp, thuật toán cho kết quả phân loại và kiểm đếm tương đối chính xác. Với mật độ lưu thông trung bình và cao, kết quả bắt đầu có độ chênh lệch và mất ổn định hơn ở các loại phương tiện là xe máy và xe tải.

Để có một kết quả cho độ chính xác cao hơn, nhóm sẽ tiến hành huấn luyện mô hình YOLOv5 với lượng dữ liệu đầu vào của xe máy, xe buýt và xe tải lớn hơn. Với tốc độ xử lý của mô hình YOLO và thuật toán Deep SORT cho tốc độ trong thời gian thực rất nhanh và ổn định với cấu hình máy phù hợp.

Các số liệu của từng loại xe lưu thông tại các thời điểm cụ thể có thể được áp dụng vào việc tính toán mật độ lưu thông trên từng khoảng thời gian, từ đó đưa ra các kết luận về mật độ lưu thông thấp, trung bình hay cao để phục vụ quá trình phân tích và điều tiết giao thông nhằm trực tiếp giảm thiểu tình trạng ùn tắc giao thông và các hậu quả của nó gây ra.

Lời cảm ơn

Nghiên cứu này là sản phẩm đồ án cuối kì của sinh viên trường đại học Công nghệ Thông tin, đại học Quốc gia Thành phố Hồ Chí Minh. Nhóm xin được gửi lời cảm ơn tới Tiến sĩ Nguyễn Lưu Thùy Ngân và Thạc sĩ Lưu Thanh Sơn – giảng viên tại trường đại học Công nghệ Thông tin, đại học Quốc gia Thành phố Hồ Chí Minh đã giúp đỡ nhóm chúng em hoàn thành được bài báo cáo này.

VII. TÀI LIỆU THAM KHẢO

[1] P. D. Khanh , "Khoa học dữ liệu - Khanh's blog," 9 Mar 2020. [Online]. Available: https://phamdinhkhanh.github.io/2020/03/09/DarknetAlgorithm.html.

[2] Sanyam, "LearnOpenCV", 21 June 2022. [Online]. Available: https://learnopencv.com/understanding-multiple-object-tracking-using-deepSORT/

[3] "Wikipedia.org," [Online]. Available: https://en.wikipedia.org/wiki/GitHub. [Accessed 10 December 2022].

[4] "RoboFlow,"[Online].Available: https://roboflow.com/. [Accessed 10 December 2022].

[5] "ACM ICPC," [Online]. Available: https://vi.wikipedia.org/wiki/ACM_ICPC. [Accessed 11 December 2022].

[6] "UIT-Together," [Online]. Available: https://uit-together.github.io/. [Accessed 11 December 2022].

[7] "RoboFlow," [Online]. Available: https://universe.roboflow.com/mark-basov/vehicles-dataset/browse?queryText=&pageSize=50&startingIndex=0&browseQuery=true. [Accessed 11 December 2022].

[8] "Github," [Online]. Available: https://drive.google.com/drive/folders/1a-v4os2Ekr-IezLE-pGNJ7R0plZyf6bE. [Accessed 11 December 2022].

[9] P. D. Khanh , "Khoa học dữ liệu - Khanh's blog," 9 Mar 2020. [Online]. Available: https://phamdinhkhanh.github.io/2020/03/09/DarknetAlgorithm.html.

[10] G. Karimi, "Section.io," 15 April 2021. [Online]. Available: https://www.section.io/engineering-education/introduction-to-yolo-algorithm-for-object-detection/.

[11] L. M. Chien, "Viblo," 4 June 2021. [Online]. Available: https://viblo.asia/p/tim-hieu-va-trien-khai-thuat-toan-non-maximum-suppression-bJzKmr66Z9N.

[12] L. M. Chien, "Viblo," 4 June 2021. [Online]. Available: https://viblo.asia/p/tim-hieu-va-trien-khai-thuat-toan-non-maximum-suppression-bJzKmr66Z9N.

[13] L. M. Chien, "Viblo," 4 June 2021. [Online]. Available: https://viblo.asia/p/tim-hieu-va-trien-khai-thuat-toan-non-maximum-suppression-bJzKmr66Z9N.

[14] J. Nelson and J. Solawetz, "YOLOv5 is Here: State-of-the-Art Object Detection at 140 FPS," 10 June 2020. [Online]. Available: https://blog.roboflow.com/yolov5-is-here/.

[15] J. Nelson and J. Solawetz, "YOLOv5 is Here: State-of-the-Art Object Detection at 140 FPS," 10 June 2020. [Online]. Available: https://blog.roboflow.com/yolov5-is-here/.

[16] A. Dadhich, Practical Computer Vision: Extract insightful information from images using TensorFlow, Keras, and OpenCV, 2018.

[17] A. Dadhich, Practical Computer Vision: Extract insightful information from images using TensorFlow, Keras, and OpenCV, 2018.

[18] "Datacamp," [Online]. Available: https://www.datacamp.com/tutorial/streamlit. [Accessed 11 December 2022].

BẢNG PHÂN CÔNG NHIỆM VỤ	
Tên thành viên	Nhiệm vụ
Nguyễn Phan Quốc Thiện - 20520775	Tìm data, gán label, xây dựng mô hình
Nguyễn Cao Quốc - 20520723	Tìm data, gán label, viết báo cáo
Đỗ Đức Thịnh - 20520780	Xây dựng demo, kiểm thử