Hệ thống giám sát giao thông thời gian thực dựa trên Deep Learning và YOLOv8

Thành viên

Cao Thị Hoài Thương

Trần Tuấn Vũ

K66A5 – Khoa học dữ liệu

Đại học Khoa học Tự nhiên, Đại học Quốc gia Hà Nội (HUS – VNU)

Table of Contents

Giới thiệu

Phân loại các phương tiện (ô tô, xe tải, xe buýt, xe máy, xe đạp, …) trên đường và thống kê thông tin về lưu lượng giao thông (ví dụ: tần suất di chuyển của các phương tiện, phương tiện đi theo hướng nào, tốc độ bao nhiêu, …) là đầu vào quan trọng trong các công cụ quy hoạch và phân tích giao thông đô thị. Tuy nhiên, giám sát lưu lượng giao thông trên đường theo thời gian thực là vấn đề đầy thách thức của các khu vực đô thị trong hiện đại với dân số ngày càng phát triển, số lượng phương tiện ngày càng tăng. Nếu quản lý giao thông đường bộ kém hiệu quả sẽ dẫn đến ùn tắc giao thông, vi phạm luật giao thông và tai nạn giao thông. Tai nạn giao thông đường bộ là một trong những nguyên nhân gây tử vong hàng đầu trên toàn cầu, vậy nên cần có những phương pháp để quản lí các phương tiện giao thông

Hiện có nhiều công nghệ phần cứng khác nhau để thu thập dữ liệu giao thông nhằm hỗ trợ các hệ thống giám sát giao thông, bao gồm cảm biến (sensors) như RADAR, LIDAR, RFID hoặc LASAR; thiết bị vòng từ (induction loops) và sóng radar (microwave radars). Nhưng tất cả các công nghệ phần cứng đều có những hạn chế:

- Các thiết bị vòng từ chỉ ảnh hưởng đến điểm đo, hạn chế phạm vi và với mật độ lưu lượng truy cập đáng kể thì độ chính xác của chúng có thể bị suy giảm.
- Phần lớn các cảm biến bề mặt có chi phí lắp đặt và bảo trì cao.
- Súng radar cầm tay hoạt động dựa trên nguyên lý hiệu ứng Doppler, ngoài giá thành thiết bị cao, cần có người điều khiển tại chỗ và sử dụng đường ngắm để thực hiện ước tính tốc độ nên chỉ có thể áp dụng cho một phương tiện tại một thời điểm
- Trong trường hợp các cảm biến không đủ, phải có người quan sá đến khu vực và đếm số lượng phương tiện đi qua.

Xét cho cùng, đây không phải là một giải pháp thực tế và những phương pháp này không thể tạo ra luồng thông tin giao thông theo thời gian thực. Tuy nhiên, các phương pháp tiếp cận trí tuệ nhân tạo (thị giác máy tính) trong thời gian gần đây, đặc biệt là các kỹ thuật xử lý hình ảnh dựa trên học máy (machine learning) và học sâu (deep learning), nằm trong số các phương pháp xử lý dữ liệu hiện đại, được sử dụng cho các hệ thống xử lý video trực tuyến.

Nghiên cứu của nhóm sẽ triển khai một hệ thống thời gian thực trong việc phát hiện các thông tin giao thông bao gồm đếm số lượng xe, phân loại xe và ước tính tốc độ xe mà không dựa trên các công nghệ phần cứng và mà sử dụng các mô hình học máy, học sâu để giải quyết các bài toán liên quan.

Cơ sở lý thuyết

Các kỹ thuật phát hiện đối tượng dựa trên mạng thần kinh tích chập và học sâu (CNN) có thể tự động trích xuất các đặc điểm từ hình ảnh đầu vào và có khả năng phục hồi tốt hơn trước những thay đổi về độ sáng tối và tình trạng che khuất một phần. Hai phương pháp phát hiện chính là phương pháp một giai đoạn (one-stage approach) và phương pháp hai giai đoạn (two-stage approach), chiếm ưu thế trong lĩnh vực phát hiện đối tượng. Các "máy dò" một giai đoạn, chẳng hạn như YOLO (You Only Look Once) hay SSD (Single-Shot Detectors), tiếp cận nhận dạng đối tượng như một bài toán hồi quy trong đó tọa độ của hộp giới hạn (bounding box) và các lớp đối tượng được dự đoán trực tiếp. Tuy nhiên, các "máy dò" hai giai đoạn, như CNN theo vùng (R-CNN), có hai giai đoạn. Bước đầu tiên là tạo ra một số lượng lớn các khu vực đề xuất. Bước thứ hai là gửi các khu vực đề xuất này để phân loại và hồi quy hộp giới hạn. Máy dò hai giai đoạn thường có tỷ lệ chính xác cao hơn máy dò một giai đoạn, nhưng vì có nhiều quy trình hơn nên chúng cũng mất nhiều thời gian xử lý hơn

Tổng quan về CNN

Còn được gọi là ConvNet, là một kiến trúc deep learning rất phổ biến, có thể học trực tiếp từ dữ liệu đầu vào mà không cần con người trích xuất một cách thủ công. Nó bao gồm nhiều lớp tích chập và gộp. CNN được thiết kế đặc biệt để xử lý nhiều hình dạng 2 chiều khác nhau và được sử dụng rộng rãi trong các ứng dụng thị giác máy tính, phân oạn hình ảnh và phát hiện đối tượng.

Sự phát triển nhanh chóng của công nghệ GPU khiến CNN trở nên phổ biến. Trên thực tế, một trong những điểm nghẽn (bottleneck) của mạng lưới thần kinh sâu (Deep Neural Networks – DNN) là việc đào tạo mất nhiều thời gian do có nhiều đơn vị ẩn trong mạng lưới. Nhưng khi GPU trở nên nhanh hơn, nút cổ chai này đã được khắc phục. Trong CNN, trạng thái của mỗi lớp được sắp xếp theo cấu trúc lưới không gian; các mối quan hệ không gian này được kế thừa từ lớp này sang lớp tiếp theo vì mỗi giá trị đặc trưng dựa trên một vùng không gian cục bộ nhỏ ở lớp trước. Mỗi lớp trong mạng tích chập là cấu trúc lưới 3 chiều có chiều cao, chiều rộng và chiều sâu. Độ sâu của một lớp đề cập đến số lượng kênh trong mỗi lớp; trong trường hợp hình ảnh RGB có màu, độ sâu là ba (R,G,B). Figure 1 dưới đây minh họa mô hình CNN truyền thống

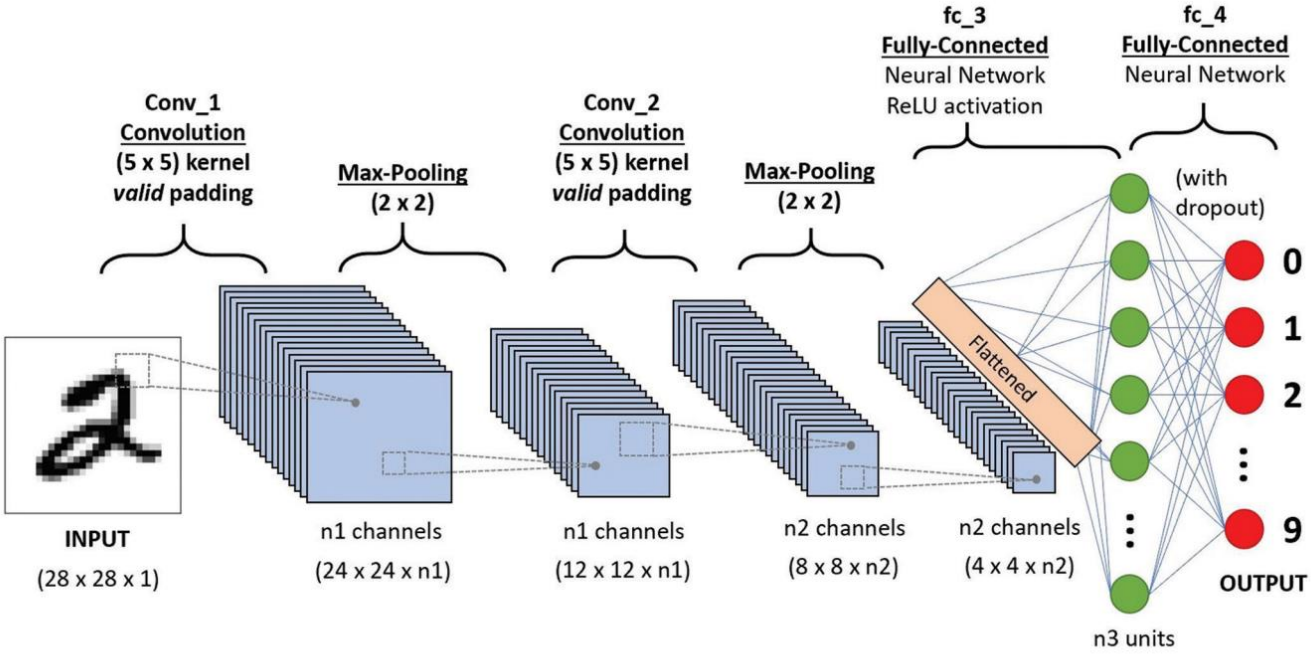
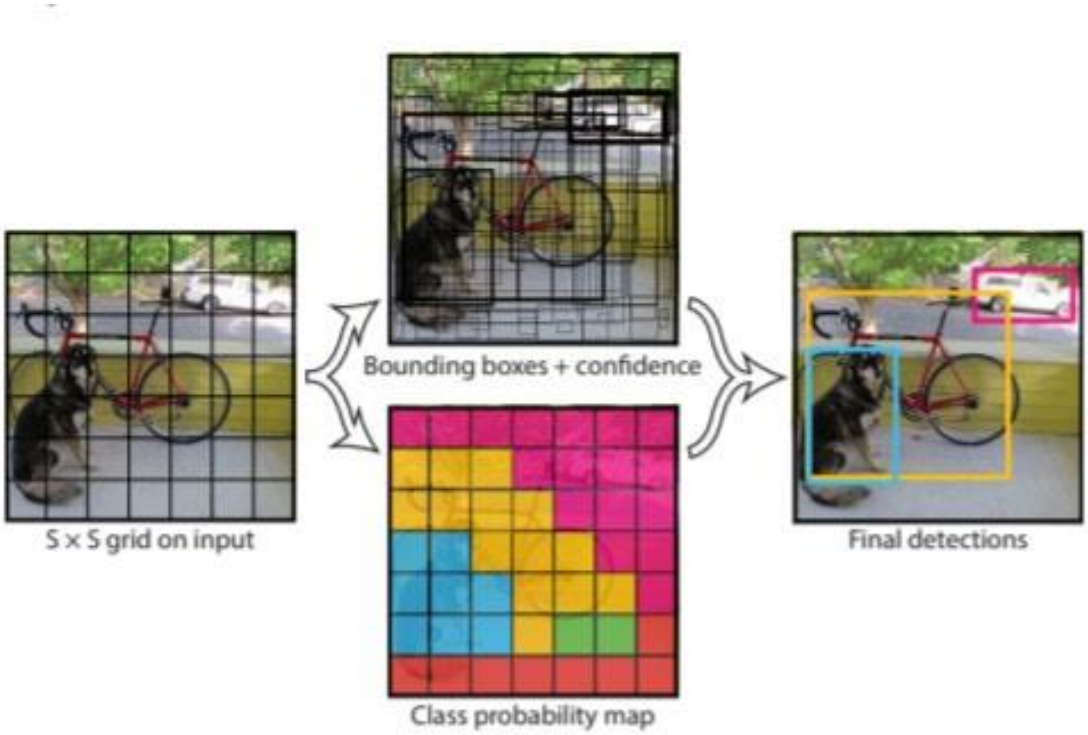


Fig. 1. Traditional CNN model. n is the number of image color channels.

Giới thiệu YOLO one-stage DNN Model

YOLO là thuật toán phát hiện đối tượng theo thời gian thực sử dụng một CNN duy nhất để dự đoán các hộp giới hạn và nhãn của các đối tượng trong ảnh.



YOLOv1 được giới thiệu vào năm 2015, coi vấn đề phát hiện đối tượng là vấn đề hồi quy thay vì vấn đề phân loại phân loại từng pixel trong hình ảnh. Để loại bỏ các phát hiện trùng lặp, YOLO chia hình ảnh đầu vào thành một lưới và dự đoán các hộp giới hạn cho cùng một lớp cùng với các giá trị độ tin cậy của nó; sau đó đầu ra được theo sau bởi mức triệt tiêu không tối đa (NMS). Có 24 lớp CNN trong kiến trúc YOLOv1, sau đó là hai lớp được kết nối đầy đủ. Tất cả các lớp đều sử dụng ReLU bị rò rỉ ngoại trừ lớp cuối cùng sử dụng chức năng kích hoạt tuyến tính. YOLO đã sử dụng các lớp chập (1*1) để giảm số lượng bản đồ đặc trưng (feature maps) và tham số mạng (network parameters). Hàm mất được sử dụng bao gồm nhiều tổng bình phương sai số (SSE).

YOLOv2 được giới thiệu vào năm 2016 với nhiều cải tiến khác nhau, bao gồm phát hiện hơn 9000 danh mục đối tượng, thêm chuẩn hóa hàng loạt, sử dụng hộp neo (anchor boxes), sử dụng kiến trúc Darknet-19 bao gồm 19 lớp tích chập và năm lớp tổng hợp tối đa.

YOLOv3 được giới thiệu vào năm 2018 và phát triển từ Darknet-19 thành Darknet-53 với các kết nối còn lại.

YOLOv4 được giới thiệu vào năm 2020 với các kết nối dư có trọng số, kết nối một phần xuyên giai đoạn (CSP), chuẩn hóa nhiều lô nhỏ, đào tạo tự đối nghịch (self-adversarial training) và chức năng kích hoạt sai làm xương sống.

YOLOv5 được giới thiệu vào năm 2021 và được triển khai theo PyTorch framework. Nó có một lớp tiêu điểm để giảm số lượng tham số và một CSP giúp mở rộng thông tin nông trong lớp tiêu điểm để tối đa hóa chức năng.

YOLOv6 được giới thiệu vào năm 2022 với đường trục đơn giản cho các mô hình nhỏ và các khối nhiều nhánh hiệu quả cho các mô hình lớn; điều này được thực hiện bằng cách đề xuất hai khung và cổ có thể điều chỉnh tỷ lệ, có thể điều chỉnh lại để phù hợp với các mô hình có kích thước khác nhau.

YOLOv7, được giới thiệu vào năm 2022, đã đề xuất một mô hình được tham số hóa lại theo kế hoạch bằng cách hợp nhất nhiều mô-đun tính toán thành một mô-đun ở giai đoạn suy luận và thực hiện một số biến đổi kiến trúc.

YOLOv8 được giới thiệu vào năm 2023 và vượt trội hơn tất cả các mô hình trước đó, như được minh họa trong Figure 2.

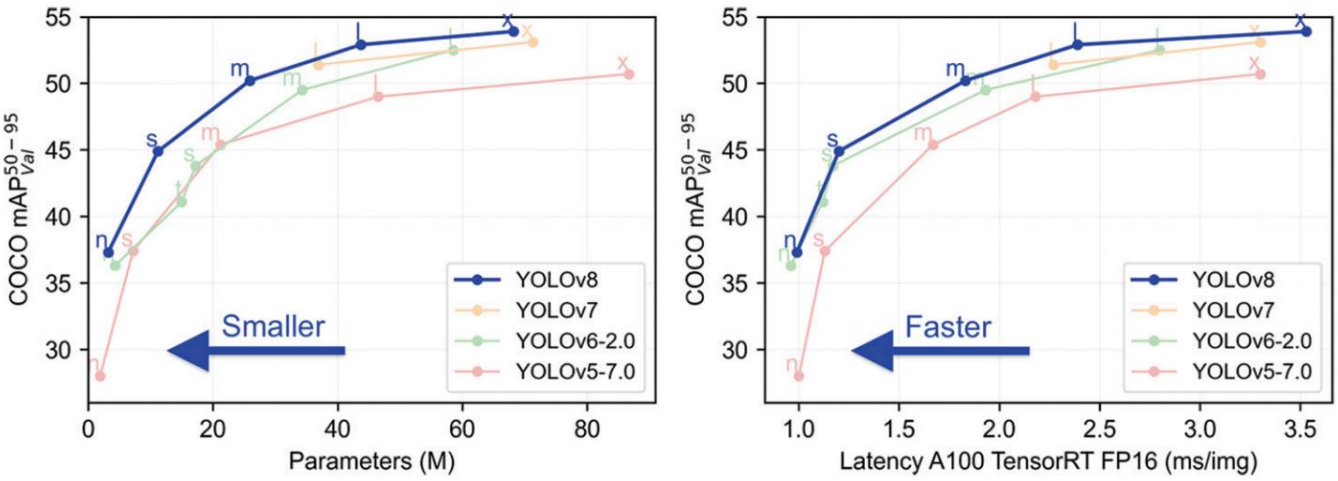


Fig. 2. YOLOv8 evaluation against recent YOLO models.

Kiến trúc YOLOv8

Là phiên bản mới nhất của hệ thống dò tìm đối tượng YOLO và đã được chứng minh là đạt được hiệu suất tiên tiến nhất (state-of-the-art) trên nhiều tiêu chuẩn khác nhau. YOLOv8 đã đề xuất một mạng đường trục mới với đầu phát hiện không có neo, nghĩa là nó dự đoán trực tiếp tâm của đối tượng thay vì phần bù từ hộp neo đã biết. Nó cũng đề xuất một chức năng mất mát mới. Kiến trúc cơ bản của YOLOv8 bao gồm hai phần chính:

Đường trục để trích xuất bản đồ đặc điểm và phần đầu để phát hiện, như được minh họa trong Figure 3. Đường trục chứa một loạt các lớp chập cho các độ phân giải và kích thước hình ảnh khác nhau, sau đó các đặc điểm được phát hiện sẽ được chuyển qua đầu nâng cao để phát hiện dựa trên hàm mất mát. Các lớp chập mới được sử dụng. Tích chập đầu tiên (6*6) của thân cây được thay thế bằng (3*3); khối xây dựng chính đã được thay đổi và C2f thay thế YOLOv5 C3. Nút cổ chai giống như trong YOLOv5, nhưng nhân của tích chập đầu tiên đã được thay đổi từ (1*1) thành (3*3) ở cổ và các tính năng được nối trực tiếp mà không buộc các kích thước kênh giống nhau. Điều này làm giảm số lượng tham số và kích thước tổng thể của tensor

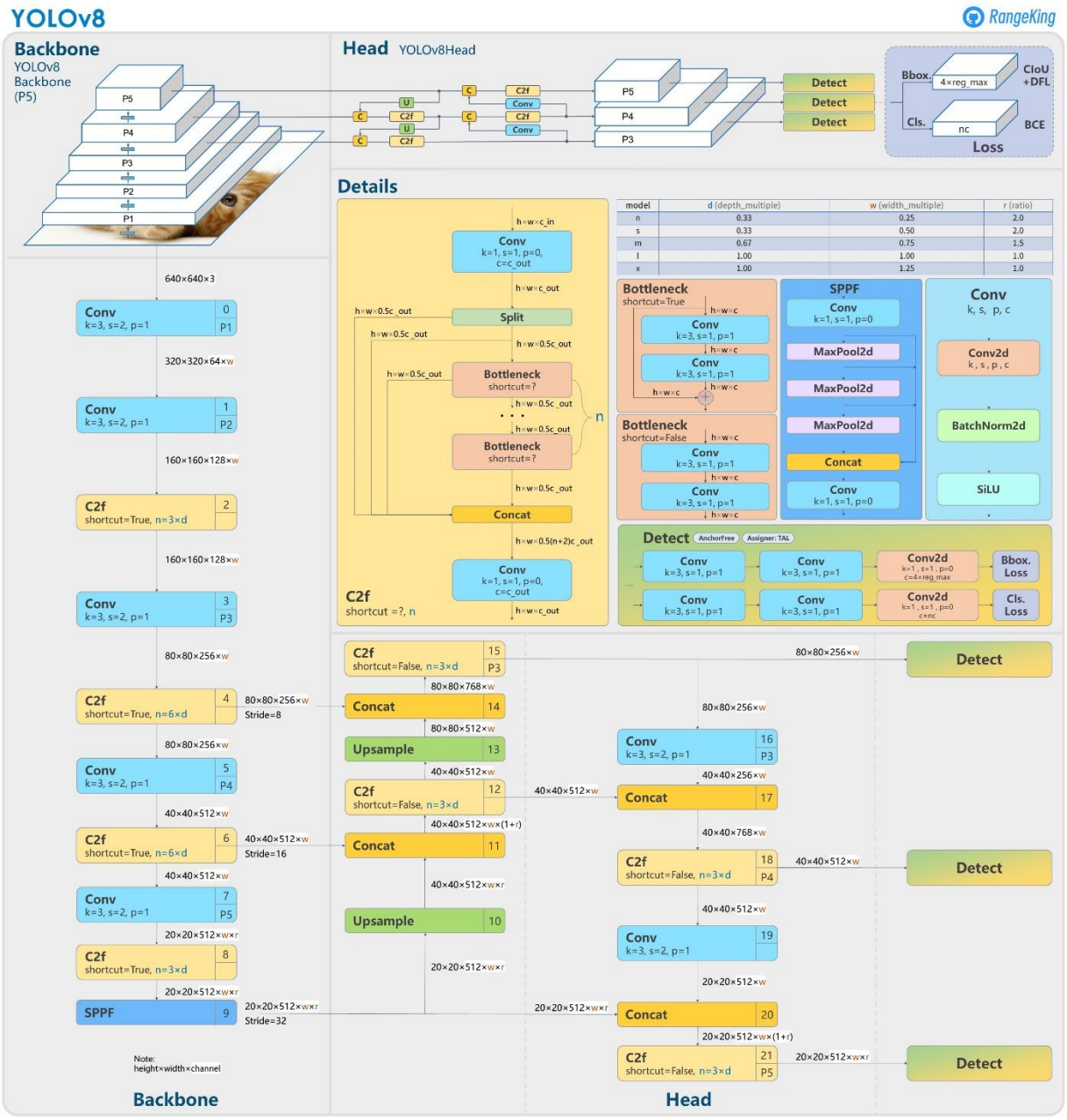


Fig. 3. YOLOv8 architecture

DeepSORT - Simple Online and Realtime Tracking with a Deep Association Metric

DeepSORT chủ yếu xác định trạng thái mới của các dấu mục tiêu và khớp kết quả phát hiện ban đầu với dự đoán bằng cách đánh giá trạng thái của các dấu. Đồng thời, các dấu có trạng thái riêng biệt có thể được đánh giá bằng số lượng khớp, tăng hoặc giảm. Quy trình thuật toán DeepSORT được thể hiện trong Figure 4.

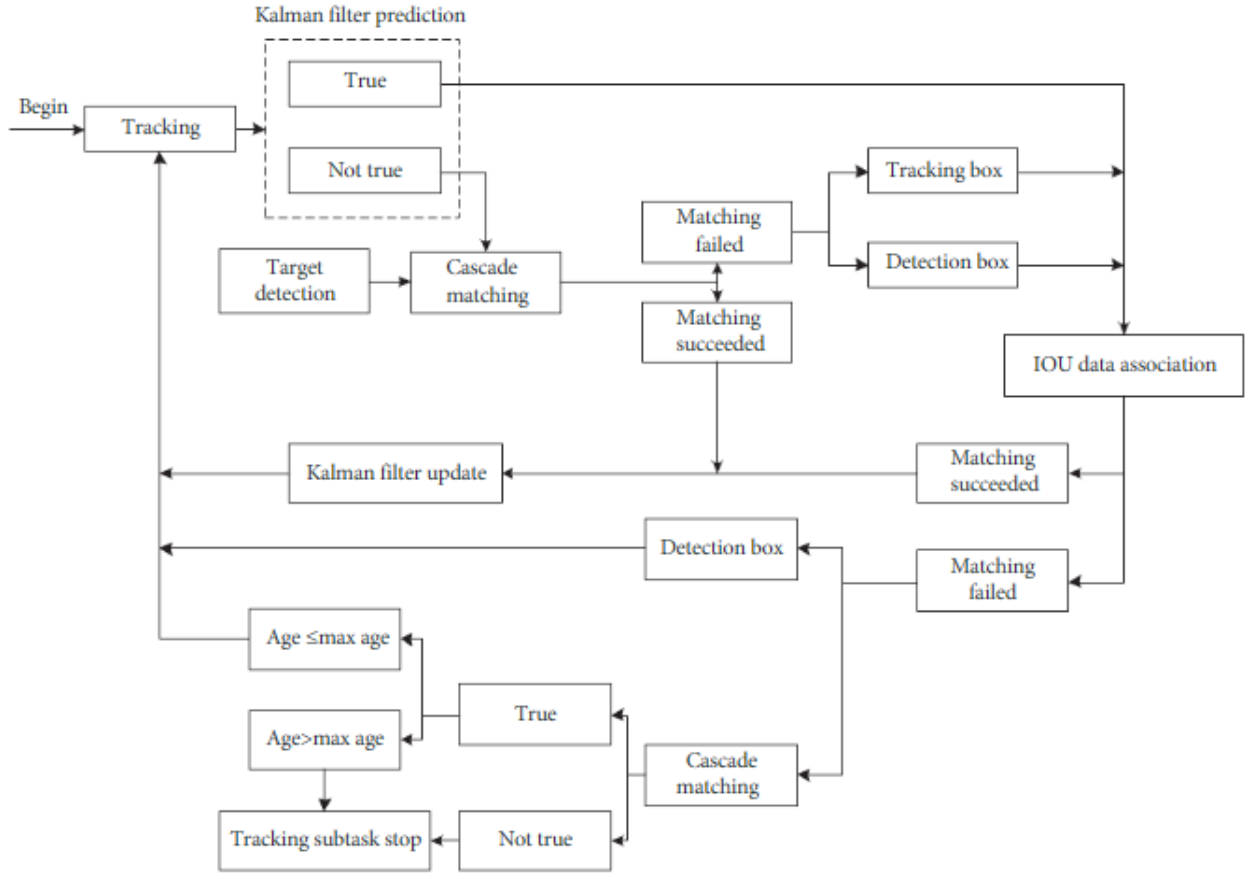


Figure 4: Block diagram of the DeepSORT algorithm

Quy trình thuật toán DeepSORT có thể được mô tả như sau

1. Phân loại Dấu (Track):
    - Dấu hiện có được phân thành hai trạng thái: xác nhận và chưa xác nhận.
2. Xếp Tầng Cho Dấu Đã Xác Nhận:
    - Dấu đã xác nhận được sắp xếp theo mức độ ưu tiên, dựa trên thời gian tồn tại và số lần khớp thành công.
    - Dấu mới khớp thành công có ưu tiên cao hơn.
3. Quy Trình Khớp Tầng:
    - Tính khoảng cách cosine và ma trận chi phí giữa đặc trưng của dấu và đặc trưng độ sâu của phát hiện.
    - Tính khoảng cách martingale giữa hộp giới hạn dự đoán và hộp giới hạn phát hiện trong chi phí ma trận.
    - Sử dụng ma trận chi phí đã tính để thực hiện ghép đôi cuối cùng thông qua thuật toán Hungary.
4. Ghép Dấu Trạng Thái Chưa Xác Nhận:
    - Tạo một tập hợp mới với các dấu trạng thái chưa xác nhận và chưa được ghép nối.
    - Sử dụng thuật toán Hungary với ma trận chi phí 1-IOU để so khớp với các hộp giới hạn phát hiện chưa được ghép nối.
5. Cập Nhật Thông Tin Bộ Lọc Kalman:
    - Các dấu được ghép nối cần cập nhật thông tin bộ lọc Kalman dựa trên thông tin của hộp giới hạn phát hiện.
6. Xác Nhận Dấu và Xóa Dấu Không Thành Công:
    - Nếu một dấu được ghép nối hơn ba lần, chuyển từ trạng thái chưa xác nhận sang xác nhận.
    - Xóa dấu chưa được xác nhận nếu số lần ghép nối không thành công vượt quá giá trị đã đặt.
    - Xóa dấu không được ghép nối thành công.
7. Tạo Dấu Mục Tiêu Mới (Nếu Cần):
    - Nếu có giới hạn phát hiện chưa được ghép nối, tạo một dấu mục tiêu mới.

Lí do chọn DeepSORT: DeepSort xuất sắc trong các cảnh đông đúc, môi trường che khuất và các kịch bản theo dõi lâu dài, nhờ vào khả năng liên kết mạnh mẽ và khả năng nhận diện lại chắc chắn vì chúng ta cần theo dõi tốc độ của phương tiện trong suốt quá trình xuất hiện trong khung hình.

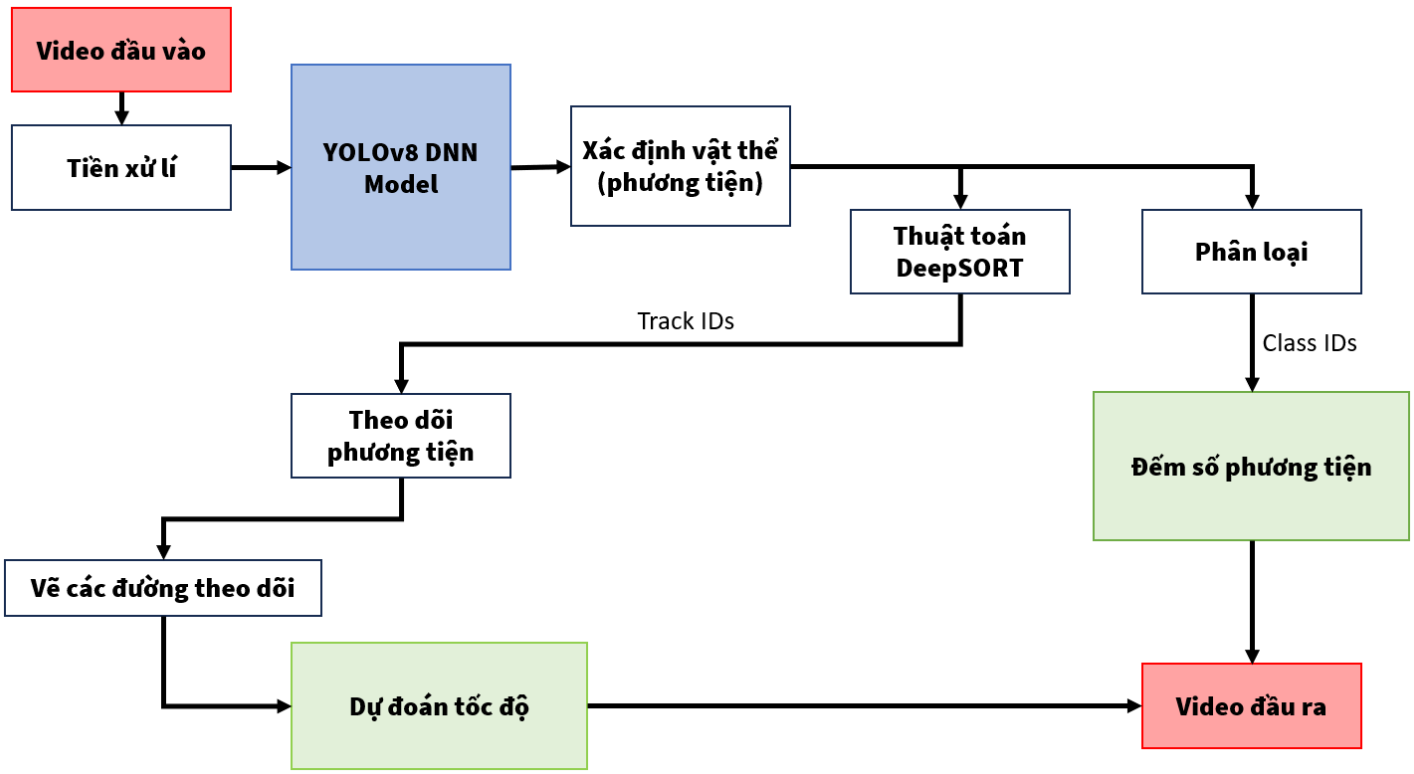
- Theo dõi đối tượng đa đối tượng trực tuyến: DeepSort xuất sắc trong việc theo dõi đa đối tượng trực tuyến, duy trì độ chính xác cao ngay cả trong các cảnh đông người nơi các đối tượng có thể chồng lấp hoặc che khuất lẫn nhau. Nó sử dụng thông tin về diện mạo, mô hình chuyển động và một phép đo khoảng cách Mahalanobis để kết hợp các phát hiện qua các khung hình.
- Khả năng nhận diện lại: DeepSort tích hợp một thành phần nhận diện lại giúp duy trì các danh tính theo dõi ngay cả khi đối tượng tạm thời biến mất khỏi tầm nhìn. Khả năng này đặc biệt hữu ích trong các tình huống bị che khuất.
- Xác nhận và sửa sai : DeepSort sử dụng một phương pháp quản lý các dấu (track) để xác nhận và sửa sai các dấu hiện tại bằng cách xem xét tính nhất quán về không gian và diện mạo của chúng. Điều này giảm thiểu lỗi dương giả (false-positive) và lỗi âm giả (false-negative) trong quá trình theo dõi.

Thiết kế thuật toán

Hệ thống được đề xuất có kiến trúc chung bao gồm các giai đoạn quan trọng. Đầu tiên, giai đoạn tiền xử lý thực hiện tính toán tỷ lệ PPM, xác định nguồn video và chia video thành các khung hình. Tiếp theo, trong giai đoạn phát hiện, sử dụng mô hình YOLOv8 đã được huấn luyện trước để phân loại và phân đoạn các loại phương tiện trong video. Sau đó, áp dụng thuật toán theo dõi để theo dõi chuyển động của các phương tiện đã được phát hiện, đồng thời gán ID cho chúng.

Giai đoạn theo dõi không chỉ quan trọng để ước tính tốc độ mà còn để đảm bảo theo dõi liên tục. Hệ thống cũng bao gồm một quy trình đếm xe, theo dõi số lượng xe đi qua đường (ROI) trong thời gian thực. Kết quả của quy trình đếm được hiển thị trực tiếp trên đầu ra video và có thể được lưu trữ dưới định dạng MP4 trên đĩa cứng.

Thêm vào đó, để tăng trải nghiệm xem, hệ thống hỗ trợ việc thử nghiệm theo dõi, tức là vẽ đường theo dõi phía sau các phương tiện để tạo cảm giác chân thực khi theo dõi. Luồng video cuối cùng có thể được trình chiếu trực tiếp cho người xem trong khi hệ thống đang hoạt động hoặc được lưu trữ lại cho mục đích ghi lại.



I. Tiền xử lí

Bước căn chỉnh camera rất cần thiết cho giai đoạn ước tính tốc độ. Việc hiệu chỉnh được thực hiện bằng cách sử dụng đối tượng tham chiếu có các kích thước đã biết (ví dụ: kích thước một ô tô đã biết).

Cần tính tỷ lệ PPM (Pixel Per Meter), dùng để ước tính khoảng cách thực tế, nó cho biết có bao nhiêu pixel trong mỗi mét của không gian thực tế. Ví dụ: chiều rộng của một chiếc ô tô là 2 mét và trong video, nó được biểu thị bằng 100 pixel, khi đó có thể nói PPM là 50. Nhưng vật thể sẽ trở nên nhỏ hơn khi càng đi xa camera. Trong trường hợp này, chỉ cần xác định khu vực tính toán PPM và việc ước tính tốc độ sẽ được thực hiện trong khu vực đã chọn trong giai đoạn ước tính tốc độ.

Ngoài ra cần phải xác định được FPS của video đầu vào, để giai đoạn ước tính tốc độ có thể sử dụng 2 thông tin này.

II. Phát hiện phương tiện

Trong giai đoạn phát hiện, chúng tôi triển khai một mô hình DNN dựa trên YOLOv8 đã được huấn luyện trước. Thuật toán phát hiện này bao gồm nhiều loại lớp:

- Lớp Conv (tích chập): Thực hiện hoạt động tích chập trên ảnh đầu vào với cửa sổ trượt, hạt nhân có kích thước xác định trước, bước trượt là 1, không phần đệm, và sử dụng hàm kích hoạt SiLU. Sau đó, áp dụng chuẩn hóa hàng loạt để cải thiện quá trình học tổng thể.
- Lớp C2f (thô đến mịn): Thực hiện tích chập với hạt nhân kích thước (1x1), không phần đệm, và bước trượt là 1. Đầu ra của nó được đưa vào thao tác phân tách, sau đó được đưa vào nút cổ chai.
- Nút cổ chai: Bao gồm hai lớp chập với các kết nối còn lại. Sau đó, thực hiện thao tác nối và sau cùng là một lớp tích chập khác.
- SPPF (tính năng tổng hợp kim tự tháp không gian): Sử dụng kim tự tháp không gian nhanh với hai lớp tích chập và ba lớp tổng hợp tối đa. Điều này giúp thu thập thông tin đa tỷ lệ, cải thiện khả năng phát hiện vật thể ở các kích cỡ khác nhau.
- Quá trình lấy mẫu (Upsample): Tăng kích thước của ma trận bằng cách sử dụng quy trình tích chập chuyển đổi với bước tiến là 2 và khoảng đệm là 1. Điều này cải thiện độ phân giải không gian của bản đồ đặc trưng, giúp định vị đối tượng chính xác hơn.

- Quá trình Concat (ghép nối): Ghép một danh sách các tensor thành một tensor một chiều. Điều này tăng tính linh hoạt và cho phép mô hình xử lý nhiều đầu vào.

III. Theo dõi phương tiện

Giai đoạn theo dõi phương tiện phụ thuộc vào kết quả của giai đoạn phát hiện, nơi các hộp giới hạn của phương tiện được xác định. Chúng tôi sử dụng thuật toán DeepSORT để theo dõi các phương tiện đang di chuyển trong video, gán một số nhận dạng (Track ID) cho mỗi phương tiện được theo dõi. Điều này là quan trọng để tính toán tốc độ của xe được theo dõi, một yếu tố quan trọng trong quá trình đếm xe.

Quá trình đếm xe bắt đầu sau giai đoạn phát hiện và trong giai đoạn theo dõi. Chúng tôi xác định một khu vực quan tâm: 1 line (ROI), nơi mà xe sẽ được đếm. Khi mỗi xe tiếp cận line thì bộ đếm phương tiện sẽ tăng lên.

Thông tin quan trọng như ID và tọa độ tâm của xe được lưu trữ trong hàng đợi. Thông tin này sẽ được sử dụng sau này trong giai đoạn ước tính tốc độ. Công thức dưới đây được sử dụng để tính tọa độ tâm của mỗi xe:

$$(cx, cy) = \frac{(x + (x + w))}{2}, \frac{(y + (h + y))}{2}$$

Trong đó:

- (x, y) là tọa độ góc trên cùng bên trái (top-left) của hộp giới hạn của đối tượng.
- (w, h) là chiều rộng và chiều cao của hộp giới hạn

IV. Ước tính tốc độ

Đích đến của việc tính tốc độ là lấy quãng đường đi được trong mỗi giây. Nhưng trong hình ảnh, không có bất kỳ thông tin nào bằng mét, do đó, cần chuyển đổi pixel di chuyển trên giây (px/s) thành mét trên giây (m/s). Sau đó từ mét trên giây (m/s) đến kilômét trên giờ (km/h).

Để tính quãng đường di chuyển, chúng ta lấy khoảng cách của cùng một đối tượng trong hai frame liên tiếp, frame hiện tại và frame trước đó. Từ phương trình toán học đơn giản, chúng ta có thể tính được khoảng cách (d_pixels) như:

$$\text{Distance in pixel} = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

Trong đó (x_2, y_2) và (x_1,y_1) tương ứng là trọng tâm của frame hiện tại và frame trước đó của xe.

Tiếp theo, tính tốc độ dựa trên công thức :

$$\text{Speed(m/s)} = \frac{\text{distance in pixels}}{\text{pixel per meter}} \times \text{frame per second}$$

$$\text{Speed(km/h)} = \frac{\text{m}}{\text{s}} \times \frac{3600\text{s}}{1000\text{m}} = \text{Speed(m/s)} \times 3.6$$

Khi đã có d_pixels và PPM được xác định từ phần tiền xử lí, chúng ta có thể thu được d_meters = d_pixels / PPM

(distance in meters = distance in pixels / pixel per meter) .

Tại sao lại nhân với FPS (frame per second) ? Bởi vì:

- Trong 1 giây có nhiều frame, tính được độ chênh lệch khoảng cách tọa độ của xe giữa frame sau và frame trước
- Nhân với tổng số frame trong 1 giây sẽ thu được tổng khoảng cách pixel xe đi được trong 1 giây (pixel/s).
- Từ khoảng cách pixel (distance in pixels) xe đi được trong 1 giây (pixel/s), đổi sang khoảng cách thực tế theo mét (distance in meters) xe đi được trong 1 giây thông qua PPM, thu được (m/s) chính là vận tốc thực của xe.
- Cuối cùng để đổi từ m/s sang km/h chúng ta sẽ nhân speed với 3.6

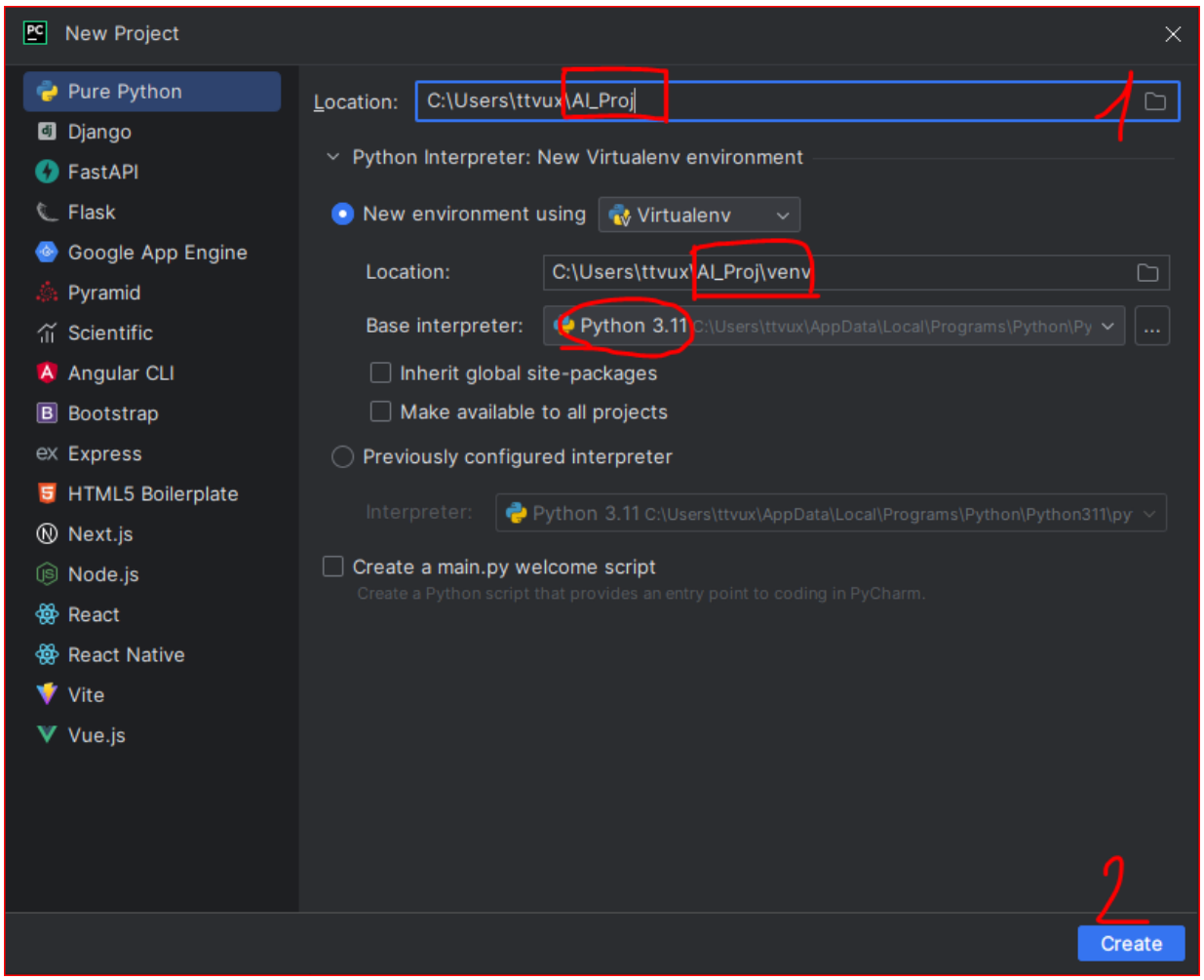
Thực thi chương trình

Link project: https://github.com/vux-66a5/HUS_K66A5_AI_Speed_Estimation

Trong dự án này, nhóm sử dụng Pycharm để thực thi code trên máy tính cá nhân và sử dụng Google Colab để thực thi online.

Sử dụng Pycharm

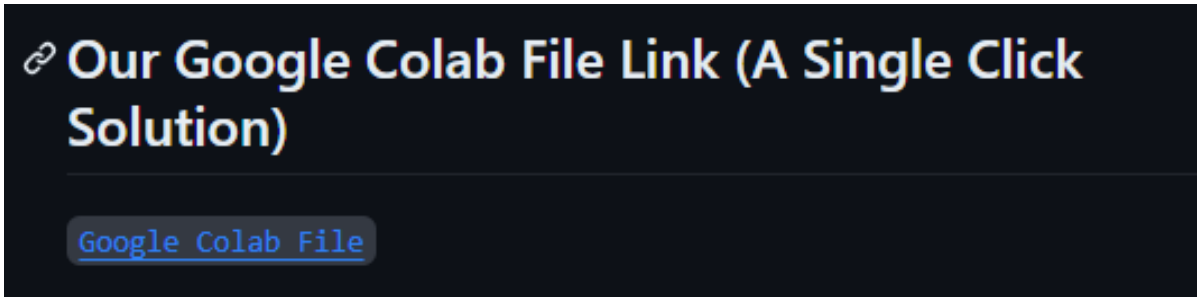
Tạo một project python trên một folder trống, tạo môi trường python ảo để tránh xung đột với môi trường python hiện có



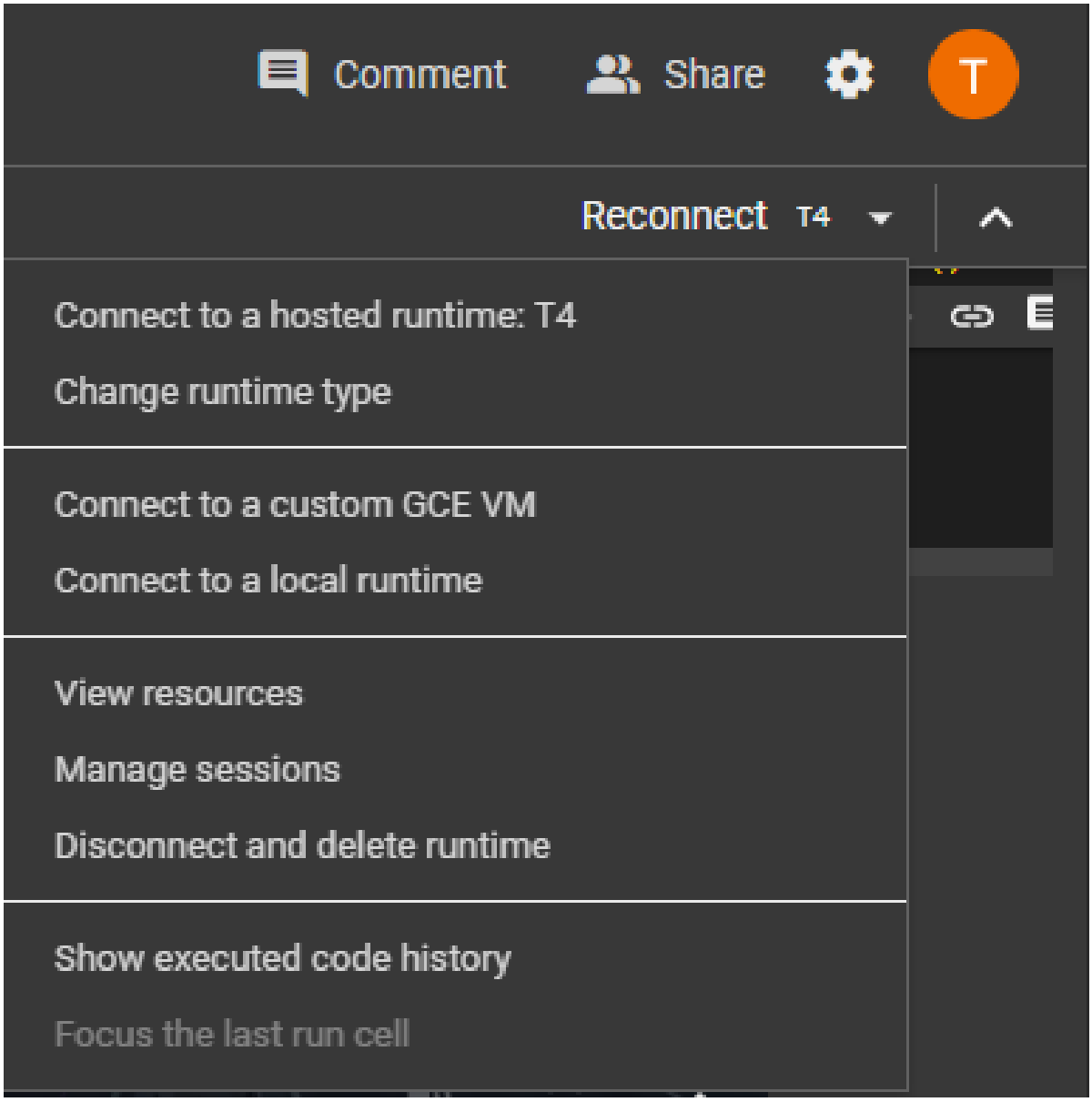
Làm theo hướng dẫn trong file README.md có trong link github

Sử dụng Google Colab

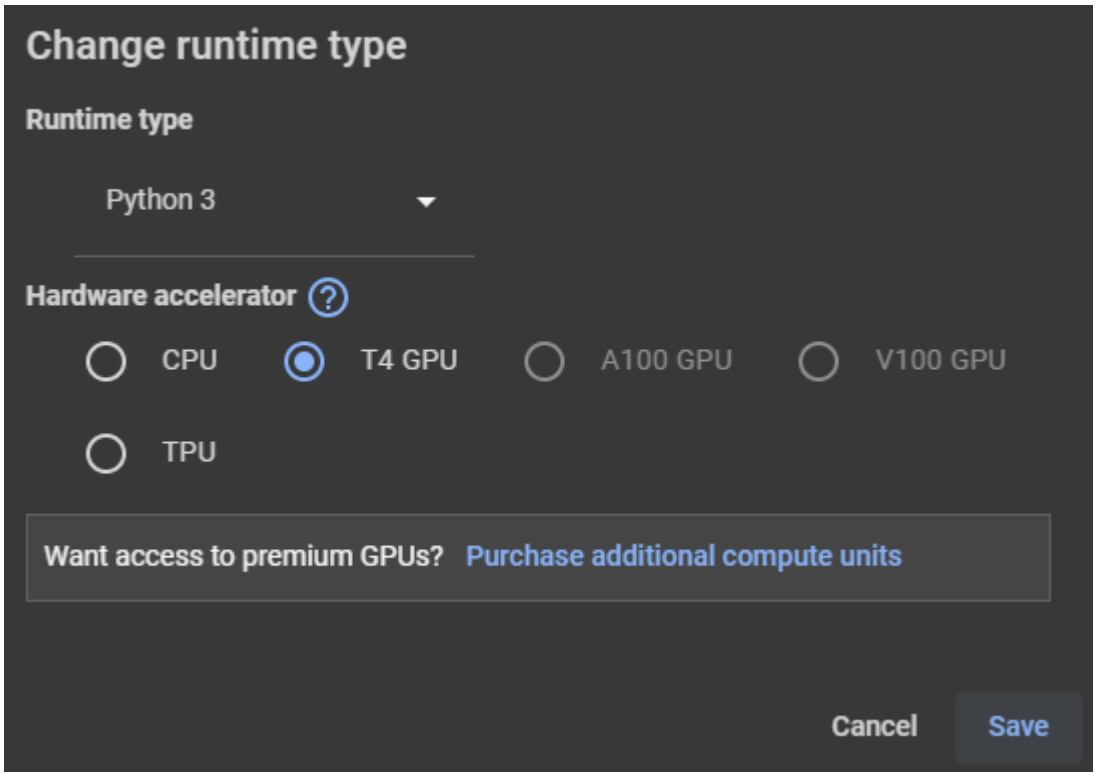
- Truy cập vào link Google Colab



- Change runtime type



- Chọn T4 GPU và Save



- Rồi lần lượt chạy từng code block đã có sẵn

Kết luận

Dự án cung cấp hệ thống giám sát giao thông trong thời gian thực bằng cách sử dụng kết hợp công nghệ học sâu và các thuật toán tiên tiến nhất (thời điểm làm dự án). Hệ thống đã cho kết quả xuất sắc ở tất cả các giai đoạn với độ chính xác cao và tỷ lệ lỗi thấp, tốc độ xử lí thời gian thực. Hệ thống được triển khai chạy trên các cấu hình phần cứng và GPU khác nhau, và có thể định cấu hình cho mọi loại đầu vào video, độ phân giải, hoặc điều kiện thời tiết.

Dự án có nhược điểm là chưa đo kích thước và khoảng cách ngoài thực tế, sai số PPM có thể dẫn đến tốc độ tính toán được không thực sự khớp với tốc độ thực tế.

Trong tương lai nhằm phát triển hơn nữa hệ thống, thuật toán phát hiện kích thước có thể được triển khai. Thông tin về kích thước xe rất hữu ích cho việc điều khiển giao thông vì giới hạn tốc độ sẽ khác nhau đối với các kích cỡ xe khác nhau. Ví dụ: xe tải có giới hạn tốc độ khác với giới hạn tốc độ của xe ô tô. Thông tin về kích thước cũng có thể giúp phát hiện các phương tiện lớn không được phép lái trên một số tuyến đường nhất định vào những thời điểm nhất định trong ngày. Chiều cao của xe cũng rất quan trọng khi lái xe trong đường hầm và dưới cầu. Ngoài ra, hệ thống có thể tích hợp thuật toán OCR để đọc biển số xe đối với xe vi phạm tốc độ và ghi lại để xử phạt.

Tài liệu tham khảo

***Each reference has a hyperlink to original source*

[1] Bashar, Saif & Karim, Abdulamir. (2023). Real-time Traffic Monitoring System based on Deep Learning and YOLOv8. ARO-The Scientific Journal of Koya University. 11. 137-150. 10.14500/aro.11327.

[2] https://khvmaths.medium.com/vehicle-speed-estimation-with-ssd-and-opencv-b512be792a4d.

[3]https://github.com/MuhammadMoinFaisal/YOLOv8_Segmentation_DeepSORT_Object_Tracking

[4] Azimjonov, Jahongir & Özmen, Ahmet. (2021). A real-time vehicle detection and a novel vehicle tracking systems for estimating and monitoring traffic flow on highways. Advanced Engineering Informatics. 50. 10.1016/j.aei.2021.101393.

[5] https://github.com/kraten/vehicle-speed-check.

[6] and other references from https://stackoverflow.com ; ChatGPT;.....