ĐẠI HỌC QUỐC GIA TP. HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC BÁCH KHOA
KHOA KHOA HỌC VÀ KỸ THUẬT MÁY TÍNH
BỘ MÔN KHOA HỌC MÁY TÍNH
----------o0o----------

ĐỒ ÁN MÔN HỌC MÁY VÀ ỨNG DỤNG

PHÁT HIỆN MŨ BẢO HỘ AN TOÀN TẠI CÔNG TRƯỜNG XÂY DỰNG SỬ DỤNG KIẾN TRÚC YOLOV5

GVHD: PGS.TS Dương Tuấn Anh
HVTH: Đặng Hiếu Ân
MSHV: 2170519

TP. HỒ CHÍ MINH, THÁNG 6 NĂM 2022

MỤC LỤC

DANH SÁCH HÌNH MINH HỌA

DANH SÁCH TỪ VIẾT TẮT

AP - Average Precision

CNN – Convolutional Neural Network

GPU - Graphics Processing Unit

ReLU - Rectified Linear Unit

ROC - Receiver Operating Characteristic

YOLO – You Only Look Once

1. MẠNG NƠ-RON TÍCH CHẬP (CNN) TRONG THỊ GIÁC MÁY

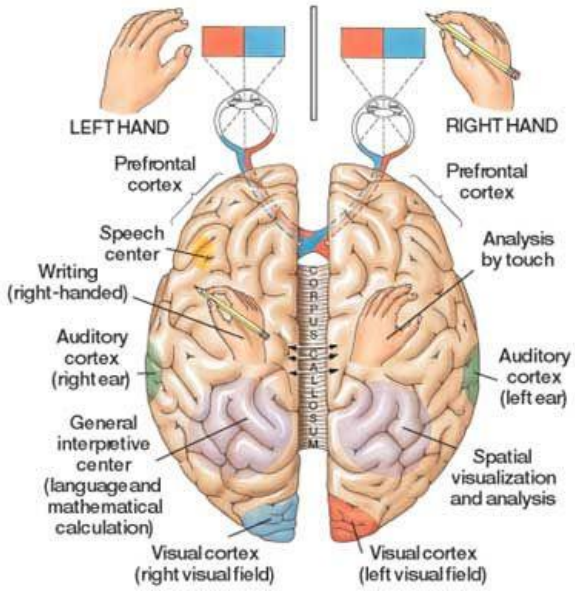
1.1 Giới thiệu CNN

Convolutional Neural Networks (CNN) là một trong những mô hình học sâu phổ biến nhất trong thị giác máy tính. CNN được dùng trong các bài toán như nhận dạng ảnh, phân tích video,….

Kiến trúc gốc của mô hình CNN xuất phát vào năm 1980 từ một nhà khoa học máy tính người Nhật. Đến năm 1998, Yan LeCun huấn luyện mô hình CNN với thuật toán lan truyền ngược (backpropagation) cho bài toán nhận dạng chữ viết tay. Đến năm 2012, Alex Krizhevsky xây dựng mô hình CNN (AlexNet) và sử dụng GPU để tăng tốc quá trình huấn luyện, kết quả đạt được top 1 trong cuộc thi thị giác máy tính ImageNet với độ lỗi phân lớp top 5 giảm hơn 10% so với những mô hình truyền thống trước đó, việc này đã tạo nên làn sóng mãnh mẽ sử dụng CNN với sự hỗ trợ của GPU để giải quyết càng nhiều các vấn đề trong thị giác máy tính.

1.2 Mối liên hệ giữa CNN và thị giác máy

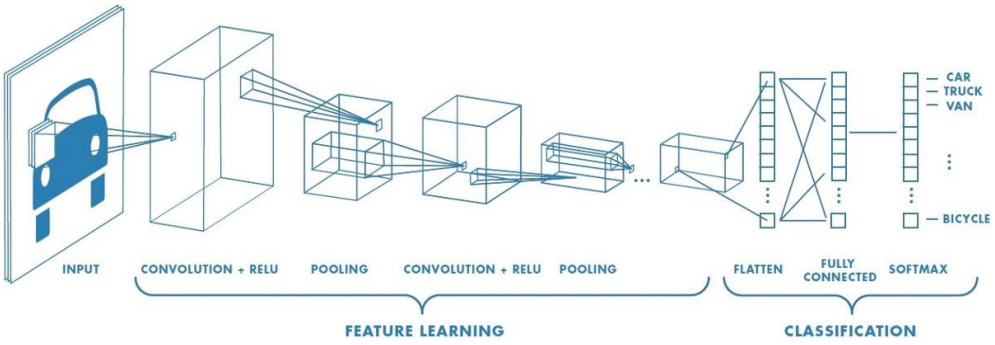
CNN có mối liên kết chặt chẽ với vỏ não thị giác của con người, vỏ não thị giác nơi xử lý thông tin liên quan đến hình ảnh từ các tế bào cảm thụ ánh sánh nằm ở mắt người. Năm 1962, hai nhà thần kinh học người Mỹ là Hubel and Wiesel đã thực hiện thí nghiệm khám phá cách tổ chức của các tế bào não để xử lý thông tin thị giác và cách các tổ chức này đảm nhận nhiệm vụ. Sau nghiên cứu này, hai nhà khoa học đã chỉ ra mỗi nơ ron được thiết lập để phản ứng lại một số đặc điểm cố định của nơ ron đó.



Hình 1. Tế bào vỏ não thị giác

1.3 Kiến trúc của CNN

CNN bao gồm tập hợp các tầng cơ bản: tầng tích chập + tầng phi tuyến, tầng gộp, tầng kết nối đầy đủ. Các tầng này liên kết với nhau theo một thứ tự nhất định. Thông thường, một ảnh sẽ được lan truyền qua tầng tích chập + tầng phi tuyến đầu tiên, sau đó các giá trị tính toán được sẽ lan truyền qua tầng gộp. Bộ ba tầng tích chập, tầng phi tuyến và tầng gộp có thể được lặp lại nhiều lần trong mạng, sau đó lan truyền qua tầng kết nối đầy đủ và hàm trung bình mũ (softmax function) để tính xác suất ảnh đó chứa vật thể gì.



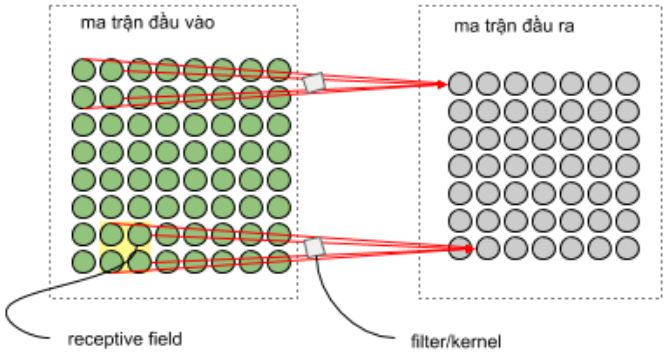
Hình 2. Kiến trúc CNN

1.3.1 Tầng tích chập

Tầng tích chập là tầng quan trọng nhất trong mô hình CNN. Chức năng của tầng tích chập là phát hiện các đặc trưng của dữ liệu đầu vào. Trong tầng này có 4 đối tượng chính là: ma trận đầu vào, bộ lọc (filters), trường tiếp nhận (receptive field), bản đồ đặc trưng (bản đồ

đặc trưng). Tầng tích chập nhận đầu vào là một ma trận 03 chiều và một bộ lọc. Bộ lọc này sẽ trượt qua từng vị trí trên bức ảnh để tính tích chập giữa bộ lọc và phần tương ứng.

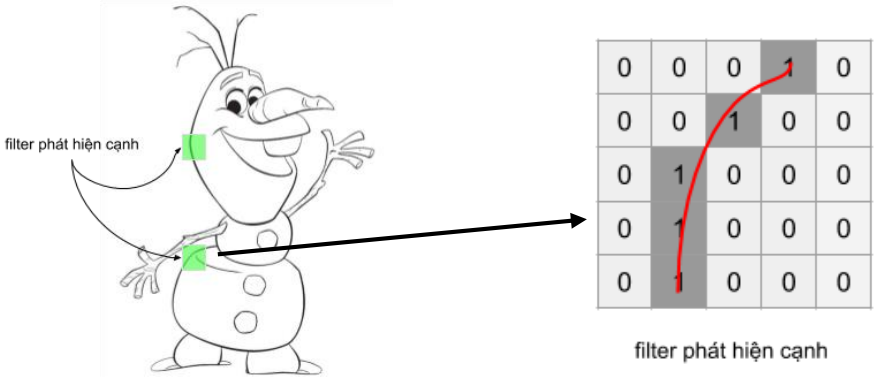
Ví dụ ở bên dưới, dữ liệu đầu vào ở là ma trận kích thước 8x8x1, một bộ lọc có kích thước 2x2x1, bản đồ đặc trưng có kích thước 7x7x1. Mỗi giá trị ở bản đồ đặc trưng được tính bằng tổng của tích các phần tử tương ứng của bộ lọc 2x2x1 với trường tiếp nhận trên ảnh.



Hình 3. Mô tả ma trận đầu vào đi qua một bộ lọc kích thước 2x2x1

Tầng tích chập có chức năng chính là phát hiện đặc trưng cụ thể của bức ảnh như màu sắc, góc cạnh, đường cong,…

Minh họa bên dưới là sử dụng một bộ lọc có kích thước 5x5 dùng để phát hiện cạnh, ta thấy được bộ lọc chỉ có giá trị1 tại các điểm tương ứng một góc cong, các giá trị xung quanh sẽ bằng 0.



Hình 4. Mô tả bộ lọc phát hiện cạnh trên hình ảnh

Kích thước bộ lọc của tầng tích chập hầu hết đều chọn số lẻ, ví dụ như 3x3 hay 5x5. Với kích thước bộ lọc lẻ, các giá trị của bản đồ đặc trưng sẽ xác định một tâm điểm ở tầng phía trước. Nếu chọn bộ lọc có kích thước chẵn sẽ gặp khó khăn khi muốn tìm vị trí tương ứng của các giá trị bản đồ đặc trưng.

Tham số stride thể hiện số pixel cần phải dịch chuyển mỗi khi trượt bộ filter qua bức ảnh.

Khi áp dụng phép tích chập thì ma trận đầu vào sẽ nhỏ dần đi làm cho số tầng của mô hình CNN sẽ bị giới hạn. Để giải quyết, cần padding vào ma trận đầu vào để đảm bảo kích thước đầu ra sau mỗi tầng tích chập là không đổi. Một cách đơn giản và phổ biến nhất để padding là sử dụng hằng số 0 như ảnh bên dưới.

0	0	0	0	0	0	0
0	1	2	6	4	9	0
0	3	5	7	9	6	0
0	4	3	4	5	1	0
0	4	8	7	9	5	0
0	5	1	6	6	5	0
0	0	0	0	0	0	0

Hình 5. Padding với hằng số 0

1.3.2 Tầng phi tuyến

Tầng phi tuyến được áp dụng vào đầu ra của các nơ-ron trong tầng ẩn của một mô hình mạng, và được sử dụng làm dữ liệu đầu vào cho tầng tiếp theo. Một số tầng phi tuyến thường gặp là: hàm sigmoid, Vanishing Gradient, ReLU,…

1.3.3 Tầng giảm chiều

Chức năng chính của tầng giảm chiều là giảm chiều của tầng trước đó, thường được sử dụng sau tầng tích chập, giúp tăng tính bất biến không gian. Tầng giảm chiều có khả năng giảm chiều, hạn chế hiện tượng quá khớp và giảm thời gian huấn luyện dữ liệu.

1.3.4 Tầng kết nối đầy đủ

Tầng kết nối đầy đủ là tầng cuối cùng trong mô hình CNN. Tầng này có chức năng chuyển ma trận đặc trưng ở tầng trước thành véc-tơ chứa xác suất của các đối tượng cần được dự đoán. Quá trình huấn luyện mô hình CNN cho bài toán phân loại ảnh cũng tương tự như huấn luyện các mô hình khác, chúng ta cần có hàm để ước tính độ lỗi của mô hình và nhãn chính xác, cũng như sử dụng thuật toán lan truyền ngược cho quá trình cập nhật trọng số.

2. GIẢI THUẬT YOLO (YOU ONLY LOOK ONCE)

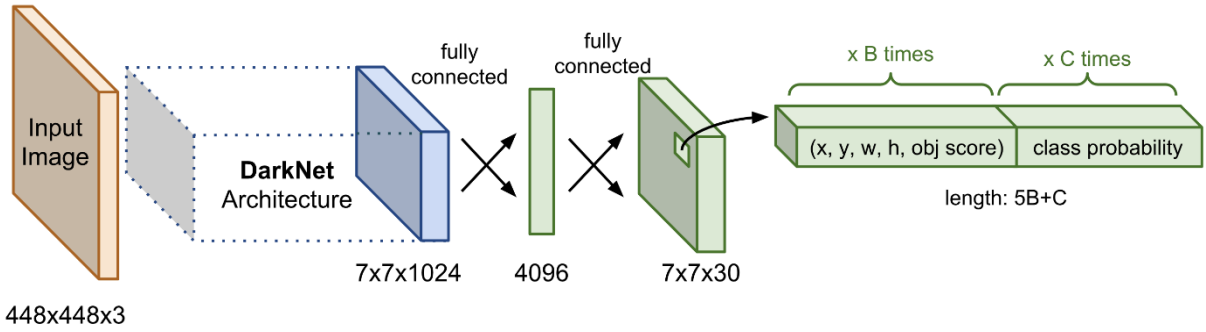
2.1 Giới thiệu mạng YOLO

YOLO là viết tắt của từ You Only Look Once, nghĩa là chúng ta chỉ cần nhìn 1 lần là có thể phát hiện ra vật thể. Xét về độ chính xác thì YOLO không phải là thuật toán tốt nhất nhưng nó là thuật toán nhanh nhất để áp dụng vào việc xử lý các bài toán vật thể detection, chính vì vậy mà nó được áp dụng rộng rãi vào các bài toán thời gian thực.

2.2 Kiến trúc mạng YOLO

Kiến trúc mạng YOLO bao gồm các mạng cơ sở là mạng tích chập làm nhiệm vụ trích xuất đặc trưng. Phần phía sau là những tầng thêm được áp dụng để phát hiện vật thể trên bản đồ đặc trưng. YOLO sử dụng chủ yếu là các tầng tích chập và các tầng kết nối đầy đủ.

Thành phần kiến trúc Darknet có tác dụng trích suất đặc trưng. Đầu ra là một bản đồ đặc trưng có kích thước 7x7x1024 sẽ được sử dụng làm đầu vào cho các lớp sau có tác dụng dự đoán nhãn và tọa độ các hộp giới hạn của vật thể.



Hình 6. Sơ đồ kiến trúc mạng YOLO

2.3 Đầu ra của YOLO

Đầu ra của mô hình YOLO là một véc tơ sẽ bao gồm các thành phần:

$$y^T = [p_0, <t_x, t_y, t_w, t_h>, <p_1, p_2, \ldots, p_c>]$$

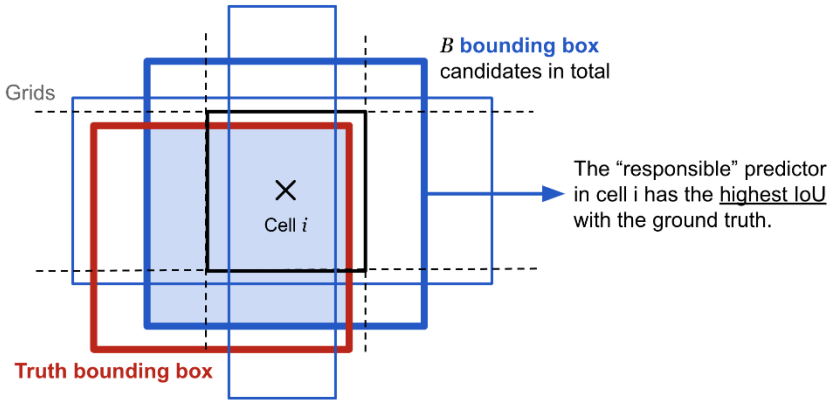
- p_0 là xác suất dự báo vật thể xuất hiện trong hộp giới hạn.
- $\langle t_x, t_y, t_w, t_h \rangle$ giúp xác định hộp giới hạn. Trong đó t_x, t_y là tọa độ tâm và t_w, t_h là kích thước rộng, dài của hộp giới hạn.
- $\langle p_1, p_2, \ldots, p_c \rangle$ là véc tơ phân phối xác suất dự báo của các nhãn lớp.

2.4 Dự báo trên nhiều bản đồ đặc trưng

Những bản đồ đặc trưng có kích thước ban đầu nhỏ giúp dự báo được các vật thể kích thước lớn. Những bản đồ đặc trưng sau có kích thước lớn hơn trong khi anchor box được giữ cố định kích thước nên sẽ giúp dự báo các vật thể kích thước nhỏ hơn.

2.5 Anchor box

Để tìm được hộp giới hạn cho vật thể, YOLO sẽ cần các anchor box làm cơ sở ước lượng. Anchor box được xác định trước và bao quanh vật thể tương đối chính xác. Mỗi một vật thể trong hình ảnh huấn luyện được phân bố về một anchor box. Trong trường hợp có từ 2 anchor boxes trở lên cùng bao quanh vật thể thì ta sẽ xác định anchor box mà có IoU với ground truth hộp giới hạn là cao nhất.



Hình 7. Mô tả cách xác định anchor box cho một vật thể

2.6 Hàm loss function

Classification Loss: lỗi của việc dự đoán loại nhãn của vật thể, hàm lỗi này chỉ tính trên những ô vuông có xuất hiện vật thể, còn những ô vuông khác ta không quan tâm. Classification Loss được tính bằng công thức sau:

$$L_{classification} = \sum_{i=0}^{S^2} \mathbb{I}_i^{obj} \sum_{c \in class} (p_i(c) - \hat{p}_i(c))^2$$

Localization loss: là hàm lỗi dùng để tính giá trị lỗi cho boundary box được dự đoán bao gồm tọa độ tâm, chiều rộng, chiều cao của so với vị trí thực tế từ dữ liệu huấn luyện của mô hình.

Giá trị hàm localization loss được tính trên tổng giá trị. Giá trị hàm lỗi dự đoán tọa độ tâm (x, y) của dự đoán hộp giới hạn và (x̂, ŷ) là tọa độ tâm của truth hộp giới hạn được tính như sau:

$$\lambda_{coord} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{obj} (x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2$$

Giá trị hàm lỗi dự đoán (w, h) của dự đoán hộp giới hạn so với truth hộp giới hạn được tính:

$$\lambda_{coord} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{obj} (\sqrt{w_i} - \sqrt{\hat{w}_i})^2 + (\sqrt{h_i} - \sqrt{\hat{h}_i})^2$$

Confidence loss là độ lỗi giữa dự đoán boundary box đó chứa vật thể so với nhãn thực tế tại ô vuông đó. Độ lỗi này tính trên cả những ô vuông chứa vật thể và không chứa vật thể.

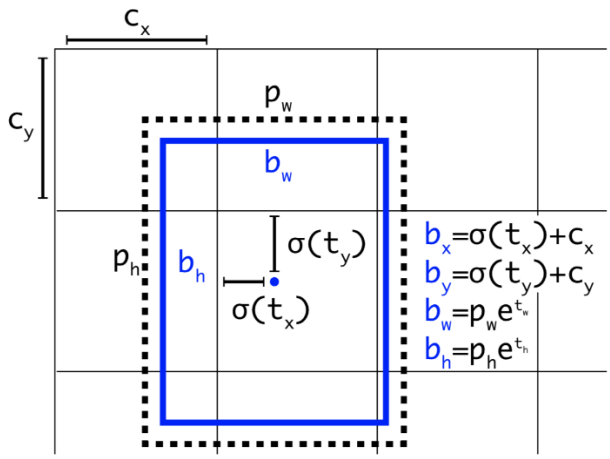
$$L_{confidence} = \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{I}_{ij}^{obj} (C_i - \hat{C}_i)^2 + \lambda_{noobject} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{I}_{ij}^{noobj} (C_i - \hat{C}_i)^2$$

Tổng lại chúng ta có hàm lỗi là tổng của 3 hàm lỗi trên:

$$L_{total} = L_{classification} + L_{localization} + L_{confidence}$$

2.7 Dự báo hộp giới hạn

Để dự báo hộp giới hạn cho một vật thể chúng ta dựa trên một phép biến đổi từ anchor box và tế bào. Cho một anchor box có kích thước (p_w,p_h) tại tế bào nằm trên bản đồ đặc trưng với góc trên cùng bên trái của nó là (c_x,c_y), mô hình dự đoán 4 tham số (t_x,t_y,t_w,t_h) trong đó 2 tham số đầu là độ lệch (offset) so với góc trên cùng bên trái của tế bào và 2 tham số sau là tỷ lệ so với anchor box. Và các tham số này sẽ giúp xác định hộp giới hạn dự đoán b có tâm (b_x,b_y) và kích thước (b_w,b_h) thông qua hàm sigmoid và hàm số mũ.



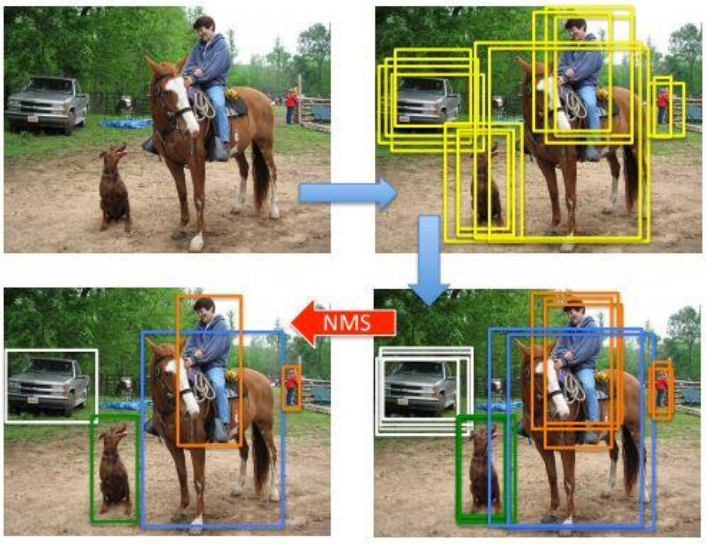
Hình 8. Mô tả dự báo hộp giới hạn trong kiến trúc YOLO

2.8 Non-max suppression

Do thuật toán YOLO dự báo ra rất nhiều hộp giới hạn trên một bức ảnh nên đối với những tế bào có vị trí gần nhau, các khung hình có khả năng bị chồng lấn lên nhau là rất cao. Trong trường hợp đó YOLO sẽ cần đến non-max suppression để giảm bớt số lượng các khung hình được sinh ra một cách đáng kể. Quá trình non-max suppression gồm 2 bước:

Bước 1: Tìm cách giảm bớt số lượng các hộp giới hạn bằng cách lọc bỏ toàn bộ những hộp giới hạn có xác suất chứa vật thể nhỏ hơn một ngưỡng threshold nào đó.

Bước 2: Non-max suppression sẽ lựa chọn ra một hộp giới hạn có xác xuất chứa vật thể là lớn nhất trong trường hợp hộp giới hạn giao nhau. Sau đó tính toán chỉ số giao thoa IoU với các hộp giới hạn còn lại.



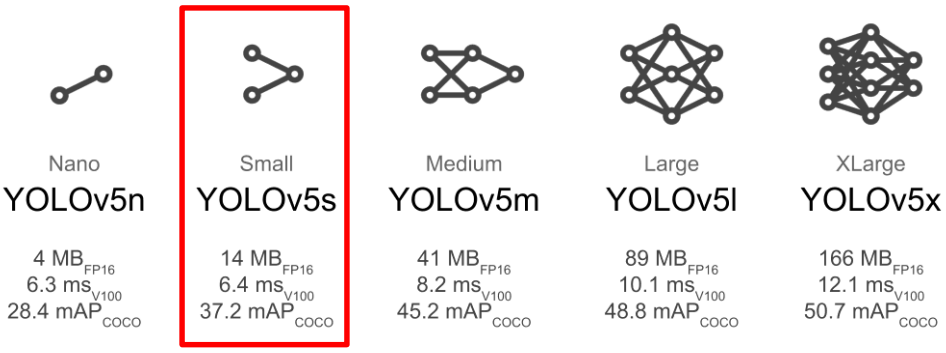
Hình 9. Mô tả của non-max suppression

3. THIẾT KẾ VÀ THỰC HIỆN MÔ HÌNH

3.1 Chọn mô hình huấn luyện

Tùy thuộc vào nhu cầu độ chính xác, thời gian xử lý của mô hình, cấu hình máy tính,…mà có thể tùy chọn phiên bản phù hợp. Ở trong khuôn khổ thực hiện đồ án này sẽ chọn mô hình là **YOLOv5s** để tiền huấn luyện dữ liệu, các phiên bản con của YOLOv5 được liệt kê chi tiết trong *Hình 10*.

- Xét về thời gian xử lý: YOLOv5s < YOLOv5m < YOLOv5l < YOLOv5x (thời gian xử lý chậm nhất)
- Về độ chính xác: YOLOv5s < YOLOv5m < YOLOv5l < YOLOv5x (độ chính xác cao nhất)



Hình 10. Lựa chọn phiên bản YOLOv5s cho việc huấn luyện

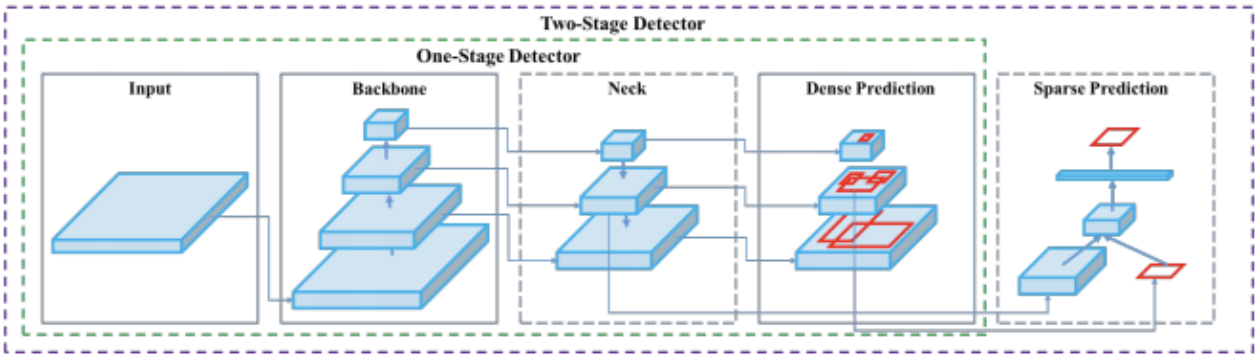
Khác với những phiên bản tiền nhiệm, kiến trúc YOLOv5 được phát triển dựa trên PyTorch thay vì DarkNet, đây là một điểm cộng rất lớn cho YOLOv5 vì PyTorch phổ biến hơn rất nhiều. YOLOv5 gồm 03 phần chính: Backbone, Head và Neck.

Phần Backbone (xương sống) là một kiến trúc học sâu hoạt động dùng để trích xuất đặc trưng của dữ liệu, tất cả các backbone về cơ bản là các mô hình phân loại. Mạng xương sống cho nhận dạng vật thể thường được đào tạo trước (tiền huấn luyện) thông qua bài toán phân loại. Tiền huấn luyện có nghĩa là trọng số của mạng đã được điều chỉnh để xác định các đặc điểm liên quan trong một hình ảnh, mặc dù chúng sẽ được tinh chỉnh trong nhiệm vụ mới là phát hiện đối tượng.

Phần Neck (phần cổ) nằm ở giữa Backbone và Head. Dùng để tổng hợp đặc trưng, đây cũng là một điểm riêng biệt của YOLO. Neck có nhiệm vụ trộn và kết hợp các bản đồ đặc trưng đã học được thông qua quá trình trích xuất đặc trưng và quá trình nhận dạng.

Phần Head như là một vật thể detector, nó sẽ tìm những vùng có khả năng chứa vật thể nhưng sẽ không xác định cho ta biết chính xác vật thể là gì, phần Head dùng để tăng khả năng phân biệt đặc trưng, dự đoán phân lớp và hộp giới hạn. Phần Head có thể sử dụng kiến trúc 1 tầng hoặc 2 tầng:

- Một tầng: gọi là Dense Prediction, có nhiệm vụ dự đoán toàn bộ hình với các mô hình như SSD, YOLO, v.v…. Nếu bài toán thuộc nhóm Dense Prediction thì nên chọn tiền huấn luyện model là Dilated Residual Networks.
- Hai tầng: gọi là Spare Prediction có nhiệm vụ dự đoán các mảng có vật thể với các mô hình thuộc các họ của Region-Based Convolutional Neural Networks (R-CNN).



Hình 11. Kiến trúc mạng YOLOv5

3.2 Chuẩn bị dữ liệu

Công việc đầu tiên là cần chuẩn bị dữ liệu cho mô hình, bộ dữ liệu cần có là dữ liệu ảnh (images) và nhãn (labels). Trong việc xây dựng mô hình thì việc chuẩn bị dữ liệu là quan trọng nhất, chúng ta không thể xây dựng bất cứ mô hình nào nếu không có dữ liệu.

3.2.1 Thu thập dữ liệu hình ảnh

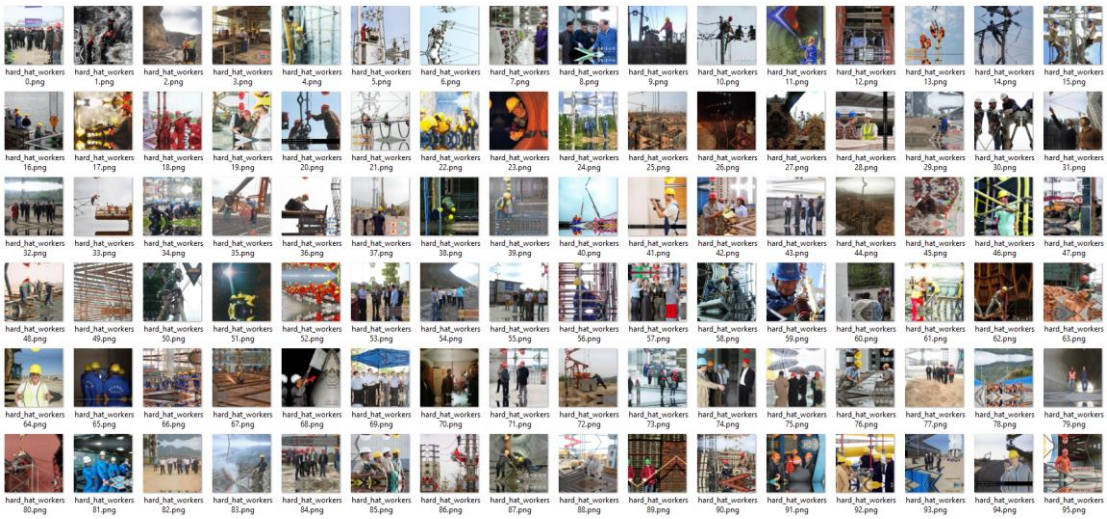
Một số phương pháp thu thập hình ảnh để sử dụng cho việc xây dựng mô hình:

- Tự chụp ảnh: phù hợp với các bài toán cần ứng dụng vào trường hợp thực tế, việc tự chụp ảnh sẽ giúp mô hình có thể học được các đặc trưng giống với bài toán đang cần giải quyết.
- Trích lọc các hình ảnh từ videos.

- Tìm với các công cụ tìm kiếm như Google hình ảnh: việc tìm dữ liệu hình ảnh bằng phương pháp này giúp chúng ta có một bộ dữ liệu hình ảnh dồi dào. Tuy nhiên việc sử dụng phương pháp này phải thận trọng vấn đề bản quyền hình ảnh.
- Các bộ dữ liệu hình ảnh được chia sẻ công khai từ các nền tảng trực tuyến của khoa học dữ liệu: có rất nhiều nền tảng trực tuyến cung cấp các bộ dữ liệu cho cộng đồng khoa học dữ liệu nghiên cứu như Kaggle, Papers with Code,…

Trong khuôn khổ của đồ án môn học này sử dụng tập dữ liệu 5000 ảnh có kích thước 416x416, gồm người có đội và không đội mũ bảo hộ tại công trường làm việc được chia sẻ công khai trên nền tảng trực tuyến Kaggle, đi kèm với tập dữ liệu hình ảnh là chú thích hộp giới hạn (hộp giới hạn annotations) ở định dạng PASCAL VOC cho 03 lớp:

- Helmet
- Person
- Head



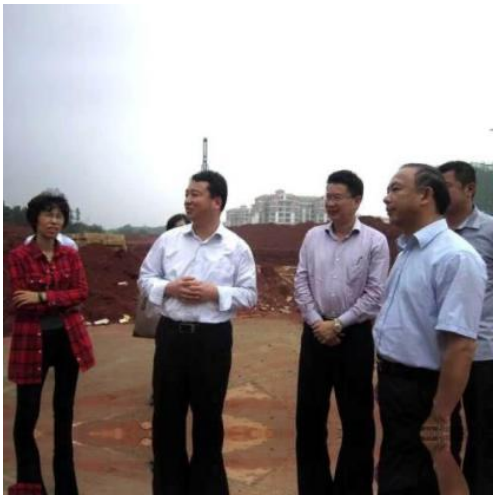
Hình 12. Bộ dữ liệu 5000 ảnh được chia sẻ từ nền tảng trực tuyến Kaggle (Nguồn:https://www.kaggle.com/datasets/andrewmvd/hard-hat-detection)

- Một số mẫu điển hình trong tập dữ liệu:

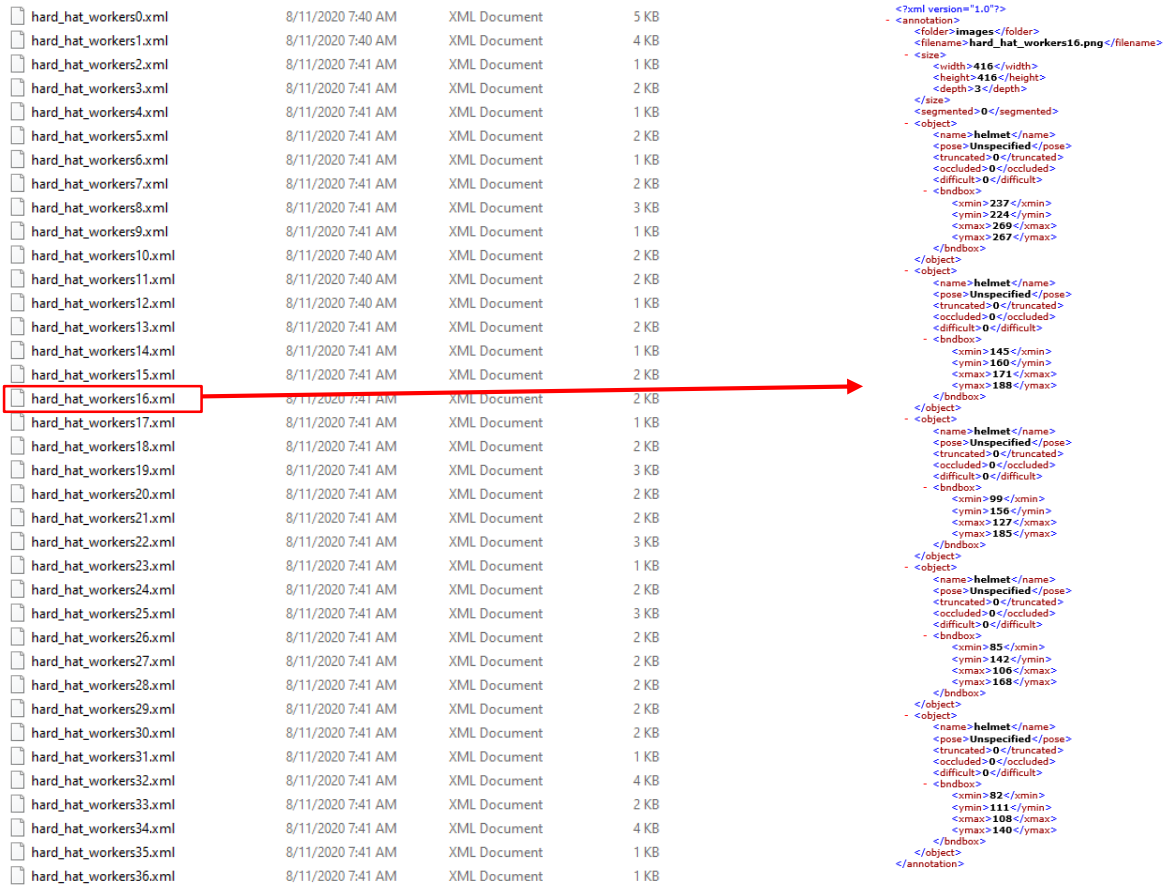
Hình 13. Mẫu (nón) có tỉ lệ lớn

Hình 14. Mẫu (nón) có tỉ lệ trung bình

Hình 15. Mẫu (nón) có tỉ lệ nhỏ



Hình 16. Mẫu âm tính (không đội mũ)

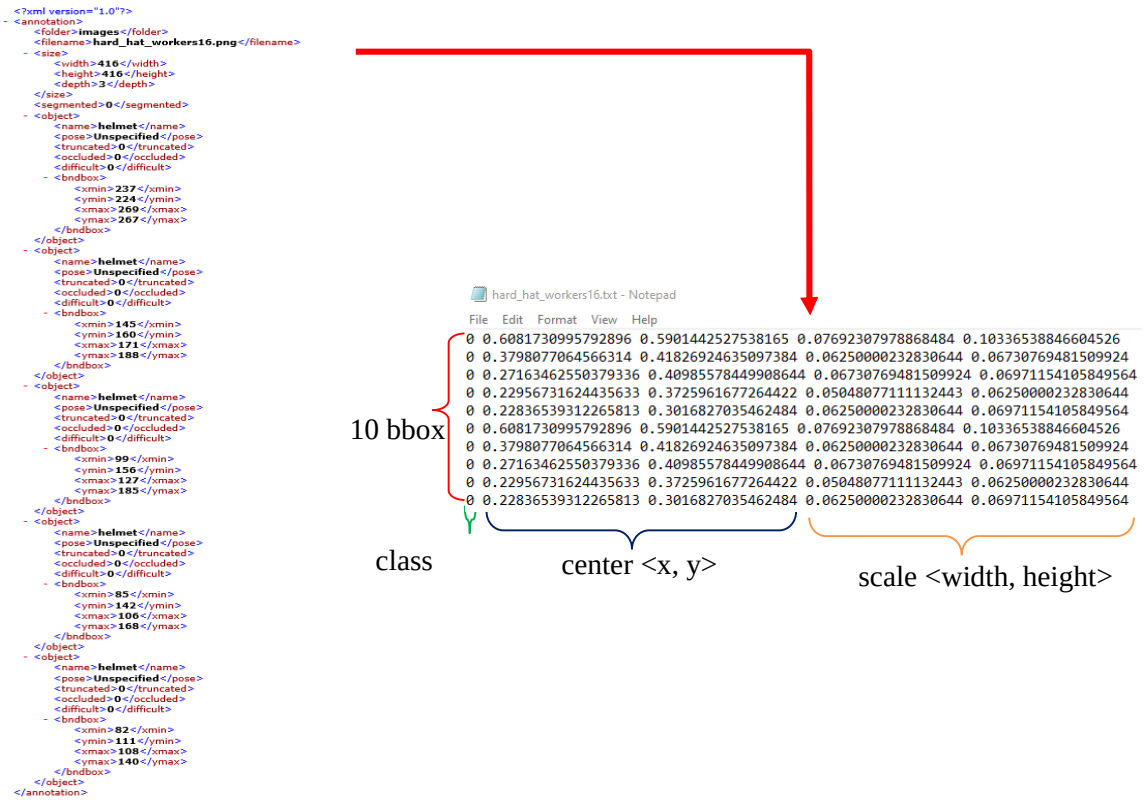


Hình 17. Chú thích hộp giới hạn (hộp giới hạn annotations) ở định dạng PASCAL VOC được cung cấp

3.2.2 Chuẩn bị dữ liệu cho huấn luyện

Vì dữ liệu đang ở định dạng .xml (được sử dụng trong hầu hết có mô hình học sâu), ta cần chuyển sang định dạng .txt (được sử dụng trong mô hình YOLO). Ở định dạng .txt, các dòng sẽ chứa 05 thông tin và cách nhau bởi dấu khoảng trắng, các tọa độ này được scale trong khoảng [0,1]

- Class index: chứa id của class mà vật thể nằm trong hộp giới hạn.
- Center <x, y>: tọa độ trung tâm theo trục tung (x) và trục hoành (y) của hộp giới hạn.
- Scale <width, height>: chiều dài và chiều cao của hộp giới hạn.

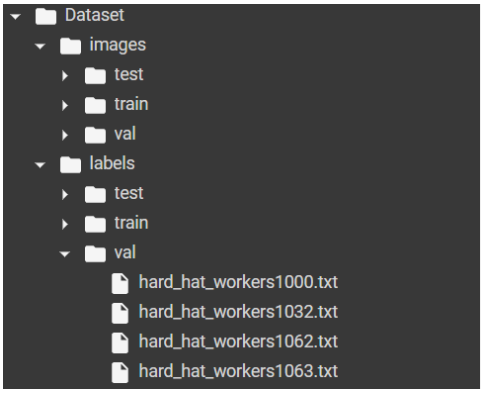


Hình 18. Chuyển đổi dữ liệu từ định dạng .xml thành định dạng .txt

3.2.3 Phân chia tập dữ liệu huấn luyện, tập kiểm chứng và tập kiểm thử (training set, validation set, testing set)

Ở bước này ta sẽ tạo 02 thư mục là images (chứa ảnh .png) và labels (chứa các tập tin .txt). Trong thư mục images chia dữ liệu ra làm 03 tập: 4000 ảnh dùng cho huấn luyện (tương ứng 4000 tập tin train.txt trong thư mục labels), 500 ảnh dùng cho kiểm chứng (tương

ứng 500 tập tin valid.txt trong thư mục labels) và 500 ảnh dùng cho việc kiểm thử (tương ứng 500 tập tin test.txt trong thư mục labels).

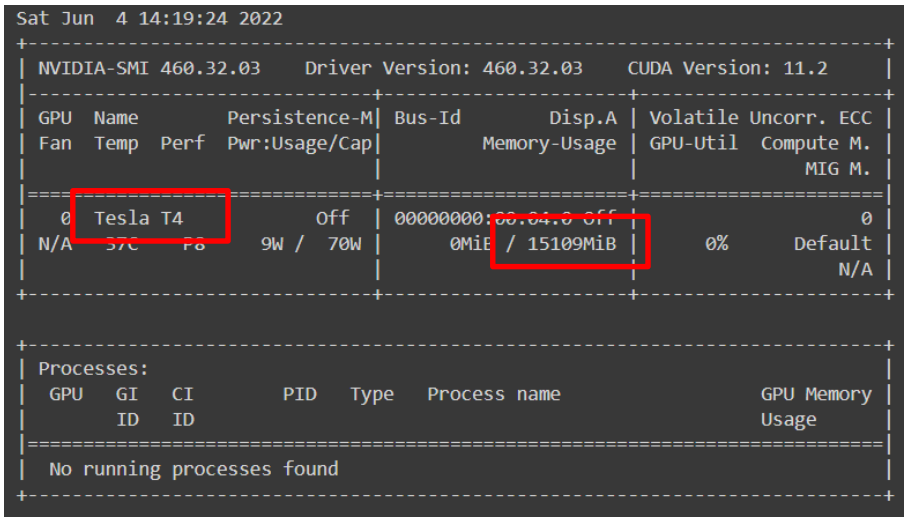


Hình 19. Phân chia dữ liệu

3.3 Huấn luyện mô hình

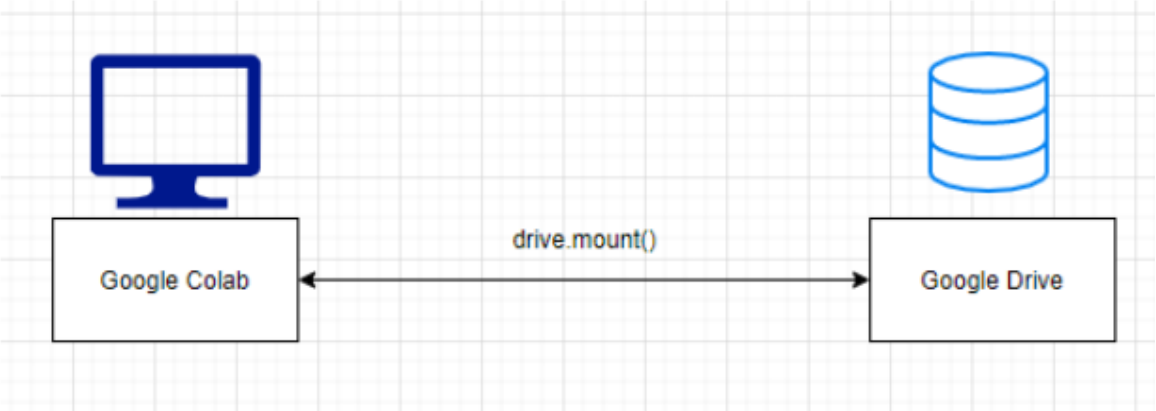
3.3.1 Google Colaboratory

Google Colaboratory là một máy đám mây ảo được cung cấp miễn phí bởi Google. Việc huấn luyện các mô hình học sâu tốn lượng tài nguyên rất lớn, việc sử dụng GPU trên cấu hình RAM của máy tính xách tay còn yếu và có thể khiến máy tính nhanh bị hư hỏng. Chưa kể việc cài đặt các framework học sâu trên máy tính rất dễ gây xung đột các package với nhau, Google Colaboratory là môi trường đã cài sẵn các packages học máy và framework học sâu thông dụng để có thể sử dụng được ngay. Mô hình này được huấn luyện với GPU Tesla T4 với bộ nhớ 15109MiB.



Hình 20. GPU Tesla T4 với bộ nhớ 15109MiB

Google Colaboratory có tác dụng như là một máy tính ảo làm nhiệm vụ tính toán, xử lý dữ liệu. Google Drive là nơi lưu trữ dữ liệu. Do đó để máy tính ảo truy cập được tới dữ liệu tại Google Drive thì ta cần phải liên kết với Drive.



Hình 21. Liên kết Google Colab với Google Drive để truy cập dữ liệu

3.3.2 Cài đặt thư viện liên quan

Tiến hành cài đặt YOLOv5 từ github về và giải nén, sau đó cài đặt các môi trường của mô hình yêu cầu:

```
# clone YOLOv5
!git clone https://github.com/ultralytics/yolov5
%cd yolov5
%pip install -qr requirements.txt  # install requirements
```

```
/content/drive/MyDrive/Helmet_detection/yolov5
```

Hình 22. Cài đặt YOLOv5 và các môi trường liên quan

3.3.3 Tiến hành huấn luyện

Mô hình YOLOv5s được huấn luyện bởi bộ dữ liệu 4000 ảnh kích thước 416x416, kích thước batch size là 64, việc huấn luyện được diễn tiến trong 50 epochs, với khoảng thời gian gần 1,3 giờ.

```
[ ]  # Train YOLOv5s
     !python train.py --img 416 --batch 64 --epochs 50 --data data/Helmet.yaml --weights yolov5s.pt
```

Hình 23. Huấn luyện với mô hình YOLOv5s

```
      Epoch   gpu_mem       box       obj       cls    labels  img_size
      47/49     7.67G   0.02874   0.02203  0.001855       236       416: 100% 63/63 [01:23<00:00,  1.32s/it]
                Class     Images     Labels          P          R     mAP@.5 mAP@.5:.95: 100% 4/4 [00:07<00:00,  1.86s/it]
                  all        500       2371      0.611      0.586      0.625      0.406

      Epoch   gpu_mem       box       obj       cls    labels  img_size
      48/49     7.67G   0.02884    0.0214  0.001617       154       416: 100% 63/63 [01:22<00:00,  1.31s/it]
                Class     Images     Labels          P          R     mAP@.5 mAP@.5:.95: 100% 4/4 [00:07<00:00,  1.94s/it]
                  all        500       2371      0.618      0.579      0.625      0.406

      Epoch   gpu_mem       box       obj       cls    labels  img_size
      49/49     7.67G   0.02826   0.02132  0.001476       298       416: 100% 63/63 [01:24<00:00,  1.34s/it]
                Class     Images     Labels          P          R     mAP@.5 mAP@.5:.95: 100% 4/4 [00:07<00:00,  1.94s/it]
                  all        500       2371      0.621      0.574      0.623      0.407

  50 epochs completed in 1.282 hours.
  Optimizer stripped from runs/train/exp/weights/last.pt, 14.3MB
  Optimizer stripped from runs/train/exp/weights/best.pt, 14.3MB

  Validating runs/train/exp/weights/best.pt...
  Fusing layers...
  Model summary: 213 layers, 7018216 parameters, 0 gradients, 15.8 GFLOPs
                Class     Images     Labels          P          R     mAP@.5 mAP@.5:.95: 100% 4/4 [00:07<00:00,  1.93s/it]
                  all        500       2371       0.61      0.587      0.627      0.407
               helmet        500       1770      0.951      0.901      0.961      0.635
                 head        500        530       0.88       0.86      0.919      0.584
               person        500         71          0          0    0.00167   0.000781
  Results saved to runs/train/exp
```

Hình 24. Kết quả huấn luyện sau 50 epochs.

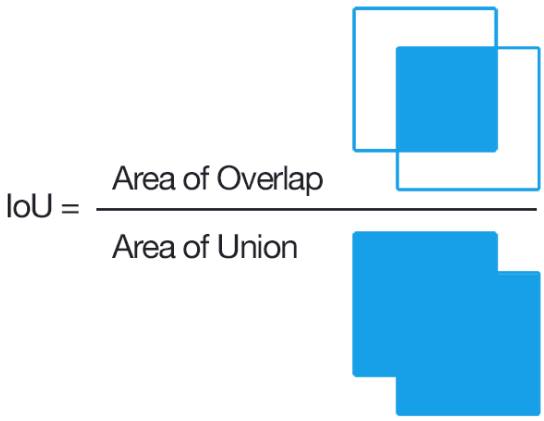
3.3.4 Phương pháp đánh giá

Các phương pháp đánh giá được sử dụng trong mô hình này:

- Tỉ lệ giao nhau (IoU): IoU = $\frac{Vùng(B_p \cap B_{gt})}{Vùng(B_p \cup B_{gt})}$

  B_p: vùng ảnh mà mô hình dự đoán đối tượng.

  B_{gt}: vùng ảnh chứa đối tượng (theo nhãn dữ liệu).



Hình 25. Mô tả công thức IoU

- Độ chính xác (Prediction): Độ chính xác = $\frac{TP}{TP+FP}$

  + Nếu True Positive (TP): Mô hình nhận dạng đúng lớp đối tượng, IoU ≥ 0.5

+ Nếu False Positive (TP): Mô hình không thể nhận dạng đúng lớp đối tượng hoặc nhận dạng đúng nhưng IoU ≤ 0.5.

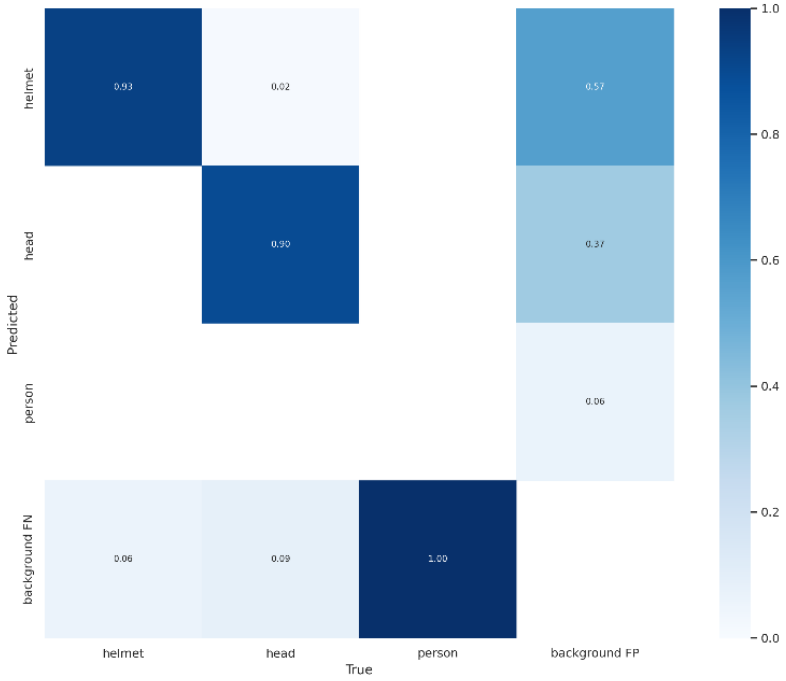
- Độ chính xác trung bình (AP): $p_{interp}(r) = max_{r' \geq r} p(r')$

  AP = $\sum_{i=1}^{n-1}(r_{i+1} - r_i)p_{interp}(r_i + 1)$

  Với r_1, r_2,….r_n là các cấp độ recall được xếp theo thứ tự tăng dần
- Độ đo COCO: mAP IoU = .50:..05..95: trung bình mAP trên các ngưỡng IoU khác nhau.

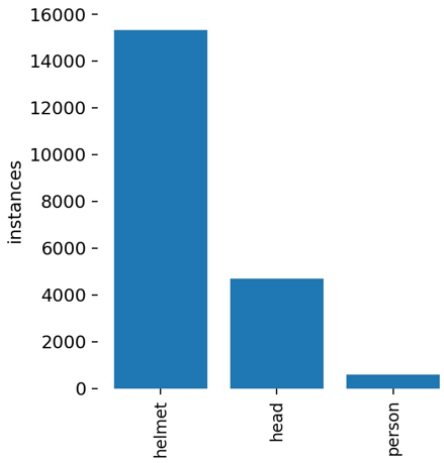
4. KẾT QUẢ VÀ PHÂN TÍCH

4.1 Đánh giá mô hình huấn luyện



Hình 26. Ma trận nhầm lẫn chuẩn hóa (normalized confusion matrix)

Ma trận nhầm lẫn chuẩn hóa thu được sau khi huấn luyện mô hình cho ta thấy được mô hình đã dự đoán khá chính xác các mẫu thuộc lớp helmet (0.93) và lớp head (0.90). Tuy nhiên, ở lớp person lại dự đoán không chính xác vì mẫu dữ liệu phân bố nhãn gán ở nhãn lớp person rất ít, điều này được thể hiện chi tiết ở *Hình 27*:



Hình 27. Chi tiết nhãn phân bố của mô hình được huấn luyện

Tuy nhiên, vì đây là bài toán nhận diện việc có đội mũ bảo hộ hay không, nghĩa là chỉ cần phân loại chính xác nhãn lớp nón bảo hộ (helmet) và đầu người (head), nên nhãn lớp người (person) không quan trọng, chúng ta sẽ tập trung vào việc đánh giá độ chính xác trên nhãn lớp đầu người (head) và nhãn lớp nón bảo hộ (helmet). Nếu đánh giá độ chính xác của mô hình trên có tính cả nhãn lớp người (person) thì sẽ có độ chính xác là:

$$\text{Độ chính xác} = \frac{TP}{TP+FP} = \frac{0.93+0.90}{0.93 + 0.90 + 1 + 0.06 + 0.09 + 0.02 + 0.57 + 0.37 + 0.06}$$

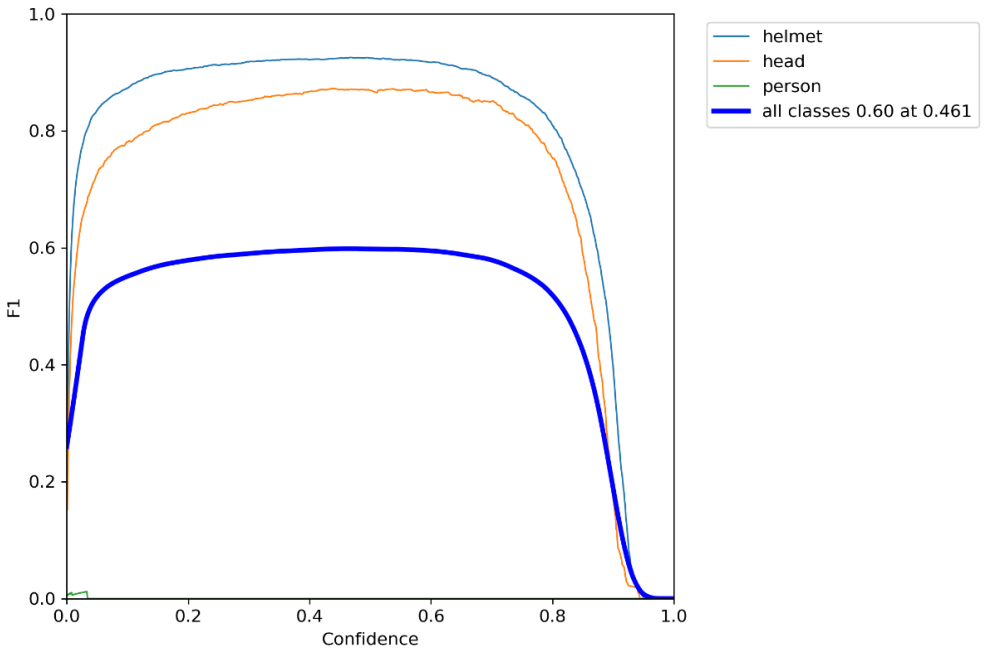
$$= 45.75\%$$

Để cải thiện ma trận nhầm lẫn này, ta có thể bỏ ra nhãn lớp người (person) và chỉ tiến hành huấn luyện với nhãn lớp là nón bảo hộ (helmet) và đầu người (head), khi đó độ chính xác có thể được cải thiện ước chừng như sau:

$$\text{Độ chính xác} = \frac{TP}{TP+FP} = \frac{0.93+0.90}{0.93 + 0.90 + 0.06 + 0.09 + 0.02 + 0.57 + 0.37 + 0.06}$$

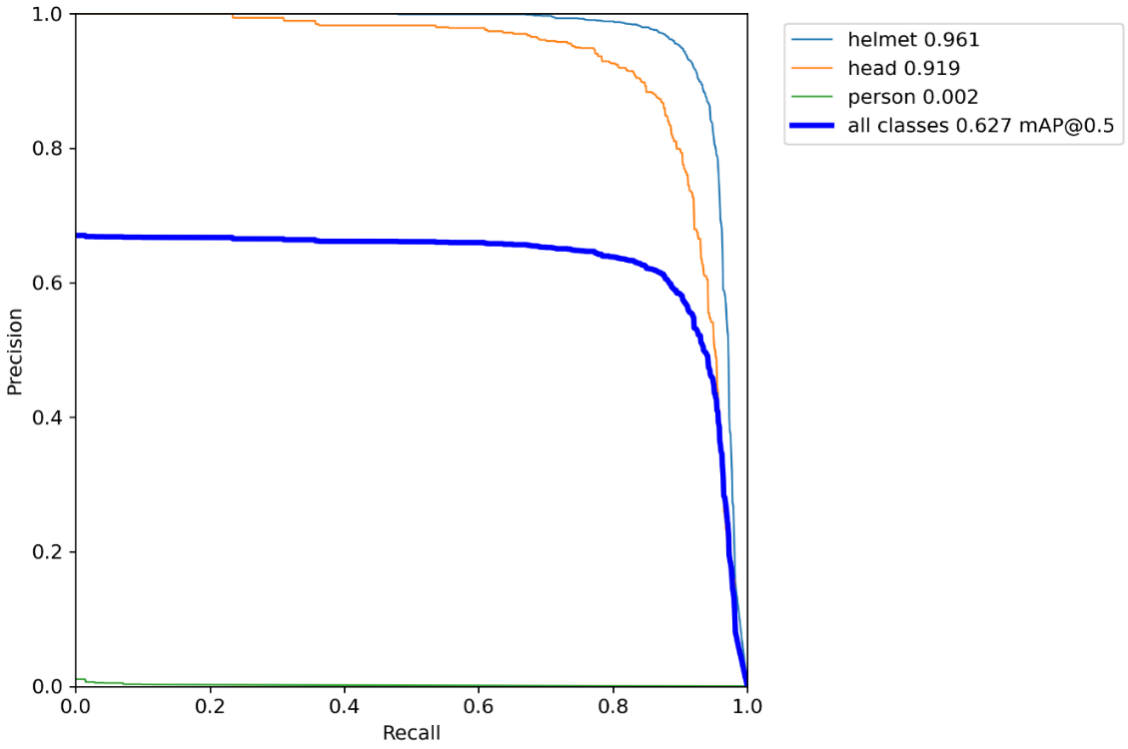
$$= 61\%$$

Như vậy, nếu chỉ huấn luyện mô hình với 02 nhãn lớp là helmet và head thì mô hình đã huấn luyện có thể đạt độ chính xác đạt khoảng 61% khi huấn luyện trong khoảng hơn 1 tiếng huấn luyện mô hình YOLOv5s với 50 epochs. Chúng ta sẽ thử nghiệm độ chính xác của mô hình này với dữ liệu thực tế ở phần tiếp theo.



Hình 28. Đường cong điểm F1 theo độ tin cậy

Đường cong điểm F1 là giá trị trung bình hài hòa (harmonic mean) của Precision và Recall, nó đánh giá độ chính xác trên đồng thời Precision và Recall, F1 score có giá trị nằm trong nửa khoảng (0, 1]. Khi F1 càng cao thì bộ phân lớp càng tốt, ngay cả khi Recall và Precision đều bằng 1 (tốt nhất có thể). Ở *Hình 28,* chúng ta thấy được đường cong helmet và head đạt điểm F1 tốt nhất tại Confidence từ khoảng [0.2; 0.7] với điểm F1 lớn hơn 0.8.

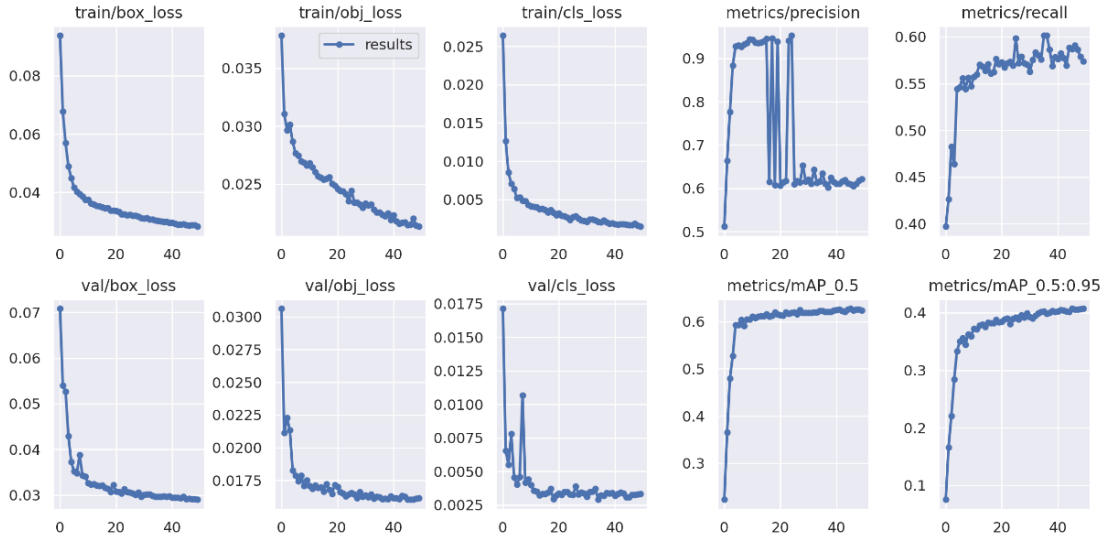


Hình 29. Đường cong PR (Precision-Recall curve)

Đường cong PR giúp chúng ta đánh giá mô hình dựa trên việc thay đổi một ngưỡng và quan sát giá trị của Precision và Recall, điều này cũng tương tự như đường cong ROC (Receiver Operating Characteristic). Như đã biết nếu giá trị ngưỡng IoU lớn hơn ngưỡng đã cho thì được coi là True Positive (nhận chuẩn), ngược lại nếu IoU bé dưới ngưỡng thì sẽ là False Positive (nhận là Positive nhưng bị sai). Từ đó, chúng ta thay đổi giá trị ngưỡng IoU, tính toán ra Precision và Recall tại các mức đó và vẽ ra được đường cong Precision Recall. Ta thấy ở *Hình 29*, helmet và head đạt được giá trị Precision lần lượt là 0.961 và 0.919 khi có Recall thấp ở khoảng 0.2. Giá trị Precision giảm dần khi Recall tăng lên, đường cong có xu hướng gần góc bên phải tương ứng phần diện tích dưới đường cong lớn, vì vậy tại các ngưỡng khác nhau thì Precision và Recall đều khá cao.

epoch	train/box_loss	train/obj_loss	train/cls_loss	metrics/precision	metrics/recall	metrics/mAP_0.5	metrics/mAP_0.5:0.95	val/box_loss	val/obj_loss	val/cls_loss	x	x	x/lr2
0	0.09384	0.037828	0.026412	0.51204	0.39707	0.22331	0.075162	0.070863	0.030629	0.017138	0.00328	0.00328	0.070476
1	0.067709	0.031056	0.012659	0.66411	0.42607	0.3657	0.1659	0.05401	0.02114	0.0065564	0.006483	0.006483	0.040345
2	0.056888	0.029624	0.0085147	0.77632	0.48241	0.4798	0.22062	0.052665	0.022274	0.0055051	0.009553	0.009553	0.010082
3	0.04895	0.030117	0.0070586	0.88308	0.46379	0.52672	0.28427	0.042935	0.021339	0.0077923	0.009406	0.009406	0.009406
4	0.044952	0.028664	0.0063851	0.9272	0.5443	0.59237	0.33321	0.037298	0.018252	0.004566	0.009406	0.009406	0.009406
45	0.028832	0.021456	0.0016608	0.61013	0.58698	0.62741	0.40692	0.029079	0.016034	0.0030929	0.001288	0.001288	0.001288
46	0.028626	0.021501	0.0016985	0.60549	0.59088	0.62338	0.40572	0.029201	0.016052	0.0032506	0.00109	0.00109	0.00109
47	0.02874	0.022028	0.0018549	0.61059	0.5864	0.62518	0.40585	0.029134	0.016056	0.0032442	0.000892	0.000892	0.000892
48	0.028838	0.021398	0.001617	0.6182	0.57875	0.62523	0.40616	0.029105	0.016125	0.0032894	0.000694	0.000694	0.000694
49	0.028259	0.021323	0.0014764	0.62136	0.57399	0.62315	0.40727	0.028999	0.016135	0.0033438	0.000496	0.000496	0.000496

Hình 30. Kết quả huấn luyện qua 50 epochs

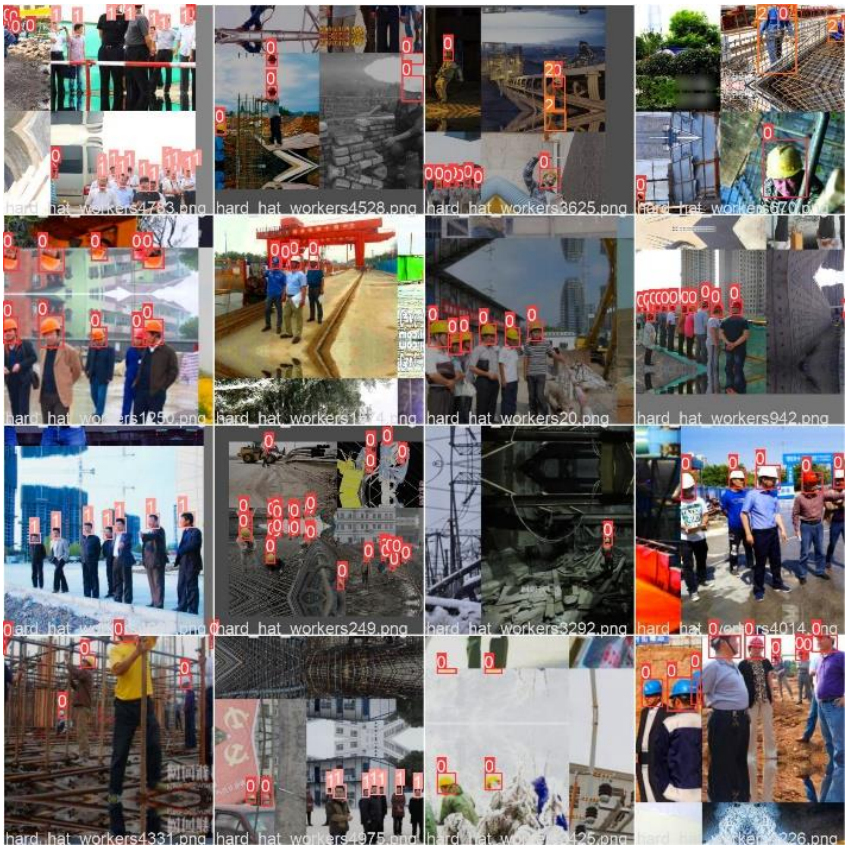


Hình 31. Trực quan hóa số liệu kết quả huấn luyện sau 50 epochs

Kết quả huấn luyện của mô hình qua từng epoch được thể hiện ở *Hình 31*. Đồ thị hàm tổn hao (loss) cho thấy thuật toán đã hội tụ sau khoảng 50 epochs đầu tiên. Hàm tổn hao ở giai đoạn sau có xu hướng tiệm cận về giá trị 0. Điều này chứng tỏ chiến lược lựa chọn hệ số học nhỏ ở 50 epochs đầu tiên đã phát huy hiệu quả và giúp thuật toán hội tụ nhanh hơn.

Giá trị mAP (Mean Average Precision) là trung bình cộng giá trị AP của các lớp khác nhau, ở mô hình này với 50 epochs huấn luyện đạt được khoảng 0.65 tại ngưỡng IoU 0.5, và khoảng 0.4 tại các ngưỡng IoU từ 0.5 đến 0.95, bước 0.05

Một số mô tả kết quả huấn luyện được trình bày ở bên dưới:



Hình 32. Nhãn huấn luyện

Hình 33. Nhãn kiểm thử

Hình 34. Nhãn hiển thị dự đoán

4.2 Dự đoán mô hình được huấn luyện với dữ liệu thực tế

4.2.1 Dự đoán với ảnh thực tế

Hình 35. Mô hình nhận dạng được ảnh người có đội và không đội mũ bảo hộ

(Nguồn ảnh: https://baobariavungtau.com.vn/kinh-te/202110/cac-cong-trinh-xay-dung-tang-toc-936216/)

4.2.2 Dự đoán với video thực tế

Hình 36. Mô hình thử nghiệm trên video đạt kết quả với độ tin cậy cao

(Nguồn video: https://youtu.be/GIgRy76TVxo)

Hình 37. Mô hình thử nghiệm trên video đạt kết quả với độ tin cậy cao

(Nguồn video: https://youtu.be/GIgRy76TVxo)

Hình 38. Mô hình có thể phát hiện được trường hợp không đội mũ trong video

(Nguồn video: https://youtu.be/2SVVsvHIu3M)

5. KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

5.1 Kết luận

Mô hình thực hiện đã đạt được độ chính xác cao khi thực nghiệm với hình ảnh vào video thực tế. Mô hình đề ra có tính ứng dụng cao trong đời sống, hoàn toàn có thể tích hợp vào hệ thống thị giác máy đặt cố định tại công trường để có thể phát hiện được các trường hợp không tuân thủ quy tắc an toàn tại nơi làm việc giúp nâng cao tính an toàn tại nơi làm việc.

5.2 Hướng phát triển

Mô hình có thể nâng cao độ chính xác bằng việc sử dụng các phiên bản YOLOv5 khác có kích thước lớn hơn, kết hợp với việc tăng cường dữ liệu đầu vào và tăng thời gian huấn luyện để cho mô hình đạt được độ chính xác cao hơn. Ngoài ra, hệ thống mô hình còn có thể phát triển vào giao thông đường bộ để phát hiện người tham gia giao thông có đội mũ bảo hiểm hay không, v.v…

6. PHỤ LỤC

6.1 Code chương trình chuẩn bị dữ liệu

```
#data download from: https://www.kaggle.com/datasets/andrewmvd/hard-
hat-detection
!cp /content/drive/MyDrive/Helmet_detection/data/archive.zip /content

#unzip data set
!unzip archive.zip

#import library
import numpy as np
import pandas as pd
from pathlib import Path
from xml.dom.minidom import parse
from shutil import copyfile
import os

#Create directories (txt files for annotation)
!mkdir -p Dataset/labels
!mkdir -p Dataset/images

#classes
classes = ['helmet','head','person']

# calculate anchor points and height width of bonding boxes
def convert_annot(size , box):
    x1 = int(box[0])
    y1 = int(box[1])
    x2 = int(box[2])
    y2 = int(box[3])

    dw = np.float32(1. / int(size[0]))
    dh = np.float32(1. / int(size[1]))

    w = x2 - x1
    h = y2 - y1
    x = x1 + (w / 2)
    y = y1 + (h / 2)

    x = x * dw
    w = w * dw
    y = y * dh
    h = h * dh
    return [x, y, w, h]

def save_txt_file(img_jpg_file_name, size, img_box):
```

```
    save_file_name = '/content/Dataset/labels/' +  img_jpg_file_name +
'.txt'
    print(save_file_name)

    with open(save_file_name ,'a+') as file_path:
        for box in img_box:

            cls_num = classes.index(box[0])

            new_box = convert_annot(size, box[1:])

            file_path.write(f"{cls_num} {new_box[0]} {new_box[1]} {new_
box[2]} {new_box[3]}\n")

        file_path.flush()
        file_path.close()

def get_xml_data(file_path, img_xml_file):
    img_path = file_path + '/' + img_xml_file + '.xml'


    dom = parse(img_path)
    root = dom.documentElement
    img_name = root.getElementsByTagName("filename")[0].childNodes[0].d
ata
    img_size = root.getElementsByTagName("size")[0]
    vật thể = root.getElementsByTagName("vật thể")
    img_w = img_size.getElementsByTagName("width")[0].childNodes[0].dat
a
    img_h = img_size.getElementsByTagName("height")[0].childNodes[0].da
ta
    img_c = img_size.getElementsByTagName("depth")[0].childNodes[0].dat
a

    img_box = []
    for box in vật thể:
        cls_name = box.getElementsByTagName("name")[0].childNodes[0].da
ta
        x1 = int(box.getElementsByTagName("xmin")[0].childNodes[0].data
)
        y1 = int(box.getElementsByTagName("ymin")[0].childNodes[0].data
)
        x2 = int(box.getElementsByTagName("xmax")[0].childNodes[0].data
)
        y2 = int(box.getElementsByTagName("ymax")[0].childNodes[0].data
)

        img_jpg_file_name = img_xml_file + '.jpg'
```

```
        img_box.append([cls_name, x1, y1, x2, y2])


    save_txt_file(img_xml_file, [img_w, img_h], img_box)

files = os.listdir('/content/annotations')
for file in files:
    print("file name: ", file)
    file_xml = file.split(".")
    print(file_xml[0])
    get_xml_data('/content/annotations', file_xml[0])

# split train val test
from sklearn.model_selection import train_test_split
image_list = os.listdir('images')
train_list, test_list = train_test_split(image_list, test_size=0.2, ran
dom_state=42)
val_list, test_list = train_test_split(test_list, test_size=0.5, random
_state=42)
print('total =',len(image_list))
print('train :',len(train_list))
print('val   :',len(val_list))
print('test  :',len(test_list))

def copy_data(file_list, img_labels_root, imgs_source, mode):

    root_file = Path( '/content/Dataset/images/'+  mode)
    if not root_file.exists():
        print(f"Path {root_file} does not exit")
        os.makedirs(root_file)

    root_file = Path('/content/Dataset/labels/' + mode)
    if not root_file.exists():
        print(f"Path {root_file} does not exit")
        os.makedirs(root_file)

    for file in file_list:
        img_name = file.replace('.png', '')
        img_src_file = imgs_source + '/' + img_name + '.png'
        label_src_file = img_labels_root + '/' + img_name + '.txt'


        # Copy image
        DICT_DIR = '/content/Dataset/images/'  + mode
        img_dict_file = DICT_DIR + '/' + img_name + '.png'

        copyfile(img_src_file, img_dict_file)
```

```
        # Copy label
        DICT_DIR = '/content/Dataset/labels/' + mode
        img_dict_file = DICT_DIR + '/' + img_name + '.txt'
        copyfile(label_src_file, img_dict_file)

copy_data(train_list, '/content/Dataset/labels', '/content/images', "tr
ain")
copy_data(val_list,   '/content/Dataset/labels', '/content/images', "va
l")
copy_data(test_list,  '/content/Dataset/labels', '/content/images', "te
st")

len(os.listdir('/content/Dataset/labels/train'))

!zip -r Dataset.zip Dataset

!cp Dataset.zip drive/MyDrive/Helmet_detection/data/
```

6.2 Code chương trình huấn luyện mô hình và dự đoán

```
#mount drive
from google.colab import drive
drive.mount('/content/drive')

#check GPU
!nvidia-smi

#unzip dataset
%cd /content
!cp /content/drive/MyDrive/Helmet_detection/data/Dataset.zip /content
!unzip Dataset.zip

cd /content/drive/MyDrive/Helmet_detection

# clone yolov5
!git clone https://github.com/ultralytics/yolov5
%cd yolov5
%pip install -qr requirements.txt  # install requirements

#import yaml
import yaml

dict_file = {'train':'/content/Dataset/images/train' ,
             'val': '/content/Dataset/images/val',
             'nc' : '3',
             'names' : ['helmet','head','person']}
```

```
with open('data/Helmet.yaml', 'w+') as file:
    documents = yaml.dump(dict_file, file)

#Tensorboard (optional)
%load_ext tensorboard
%tensorboard --logdir runs/train

#Train with YOLOv5s
!python train.py --img 416 --batch 64 --epochs 50 --
data data/Helmet.yaml --weights yolov5s.pt

#test model
!python detect.py --
weights /content/drive/MyDrive/Helmet_detection/yolov5/runs/train/exp/w
eights/best.pt --source '/content/drive/MyDrive/Helmet_detection/2.mp4'
```

7. TÀI LIỆU THAM KHẢO

[1] PGS.TS Dương Tuấn Anh, Chương 12 Phần I –Học sâu với mạng thần kinh nơ ron, Học Máy và Ứng Dụng, Khoa Khoa Học và Kỹ Thuật Máy Tính, Đại Học Bách Khoa TPHCM.

[2] PGS.TS Dương Tuấn Anh, Chương 9 – Sự kết hợp của các bộ phân lớp, Học Máy và Ứng Dụng, Khoa Khoa Học và Kỹ Thuật Máy Tính, Đại Học Bách Khoa TPHCM.

[3] Glenn Jocher, Github, https://github.com/ultralytics/yolov5

[4] Larxel, Kaggle, https://www.kaggle.com/datasets/andrewmvd/hard-hat-detection

[5] Phạm Đình Khánh, Khoa học dữ liệu blog, https://phamdinhkhanh.github.io/2020/03/09/DarknetAlgorithm.html

[6] Quoc Pham, Khoa học dữ liệu blog, https://pbcquoc.github.io/yolo/

[7] Vũ Hữu Tiệp, Diễn đàn Machine Learning cơ bản, https://machinelearningcoban.com/2017/08/31/evaluation/

[8] Redmon, J., & Farhadi, A. (2018). YOLOv3: An Incremental Improvement. *ArXiv, abs/1804.02767*.

[9] Redmon, J., Divvala, S.K., Girshick, R.B., & Farhadi, A. (2016). You Only Look Once: Unified, Real-Time Object Detection. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 779-788.