

ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC KHOA HỌC XÃ HỘI VÀ NHÂN VĂN
KHOA THƯ VIỆN - THÔNG TIN HỌC



BÁO CÁO ĐỒ ÁN
MÔN: LƯU TRỮ VÀ KHAI THÁC DỮ LIỆU
ĐỀ TÀI
NHẬN DIỆN ĐỐI TƯỢNG BẰNG
THUẬT TOÁN YOLOv5

GVHD: Ths. Nguyễn Tấn Công

Họ và tên: Lưu Hải Phú

MSSV: 22562100046

Lớp: Quản lý Thông tin – B

TP. Hồ Chí Minh, ngày 17 tháng 1 năm 2025

MỤC LỤC

DANH MỤC HÌNH ẢNH.....	2
DANH MỤC BẢNG.....	3
CHƯƠNG I: GIỚI THIỆU ĐỀ TÀI.....	5
1.1. Lý do chọn đề tài.....	5
1.2. Mục tiêu của đồ án.....	5
1.3. Giới hạn đề tài.....	5
1.4. Phương pháp nghiên cứu	6
1.5. Phạm vi thực hiện	6
CHƯƠNG II: CƠ SỞ LÝ THUYẾT	6
2.1. Tổng quan về YOLO.....	6
2.1.1. Lịch sử phát triển	6
2.1.2. So sánh các mô hình.....	7
2.1.3. Kiến trúc mạng YOLO.....	10
2.1.4. Nguyên lý hoạt động của mạng YOLO.....	12
2.2. Output của YOLO.....	13
2.2.1. Dự báo trên nhiều feature map.....	14
2.2.2. Anchor box.....	15
2.2.3 Hàm mất mát (Loss Function)	17
2.3. Dự báo bounding box.....	19
2.3.1. Non-max suppression.....	21
2.4. Chi tiết về thuật toán YOLOv5	21
2.4.1. Khái niệm.....	21
2.4.2. Kiến trúc thuật toán YOLOv5	22
2.4.3. Quá trình hoạt động.....	22
2.4.4. Phân loại YOLOv5.....	23
2.4.5. Nguyên lý hoạt động của YOLOv5.....	24
CHƯƠNG 3: CHUẨN BỊ DỮ LIỆU	34
3.1. Thu thập dữ liệu	34
3.2. Khám phá và đánh giá dữ liệu	36
3.3. Làm sạch và xác thực dữ liệu.....	36
3.4. Lưu trữ dữ liệu	36
CHƯƠNG 4: CÀI ĐẶT CHƯƠNG TRÌNH.....	37
4.1. Quá trình huấn luyện đối với dữ liệu lớn thu thập	37

4.2. Quá trình huấn luyện đối với dữ liệu cá nhân	39
CHƯƠNG 5: THIẾT KẾ HỆ THỐNG	43
5.1. Thiết kế website nhận diện.....	43
5.2. Tạo demo webcam nhận diện.....	46
CHƯƠNG 6: BÁO CÁO KẾT QUẢ	47
6.1. Tổng kết kết quả thực nghiệm.....	47
6.1.1. Mô tả quá trình thực nghiệm.....	47
6.1.2. Bảng tổng hợp kết quả thí nghiệm khi train 2 dataset.....	47
6.1.3. Đánh giá hiệu quả thuật toán.....	47
6.2. Nhận xét và đánh giá.....	48
6.2.1. Điểm mạnh của thuật toán YOLOv5.....	48
6.2.2. Điểm yếu của thuật toán YOLOv5.....	48
6.2.3. Đề xuất cải tiến và mở rộng	48
6.3. Kết luận và hướng phát triển.....	48
6.3.1. Kết luận.....	48
6.3.2. Hướng phát triển	49
6.3.3. Hạn chế của đề tài	49
TÀI LIỆU THAM KHẢO	50

DANH MỤC HÌNH ẢNH

Figure 1. So sánh các phiên bản YOLOv5.....	7
Figure 2. Kiến trúc mạng YOLO	10
Figure 3. Các layer trong mạng darknet-53	11
Figure 4. Cách hoạt động của mạng YOLO.....	12
Figure 5. Kiến trúc một output của model YOLO	13
Figure 6. Các feature maps của mạng YOLOv3 với input shape là 416x416, output là 3 feature maps có kích thước lần lượt là 13x13, 26x26 và 52x52.	14
Figure 7. Xác định anchor box cho một vật thể.....	16
Figure 8. Khi 2 vật thể người và xe trùng mid point và cùng thuộc một cell. Thuật toán sẽ cần thêm những lượt tiebreak để quyết định đâu là class cho cell.	17
Figure 9. Tính toán Loss Function cho 2 object: tam giác và hình thoi.....	18
Figure 10. Công thức ước lượng bounding box từ anchor box.....	20
Figure 11. Non-max suppression. Từ 3 bounding box ban đầu cùng bao quanh chiếc xe đã giảm xuống còn một bounding box cuối cùng.....	21
Figure 12. Kiến trúc của YOLOv5.....	22
Figure 13. Quá trình nhận diện đối tượng của YOLOv5	23
Figure 14. Các phiên bản của YOLOv5.....	24

Figure 15. So sánh tốc độ xử lý và độ chính xác giữa các phiên bản	24
Figure 16. Ảnh sau khi tính resize và cells	30
Figure 17. Kết quả minh họa của tìm bounding box.....	34
Figure 18. Link tải dataset và số lượng đối tượng nhận diện trên Github	35
Figure 19. Ảnh khuôn mặt tự tạo	35
Figure 20. Nhãn được gán của ảnh khuôn mặt	35
Figure 21. Gắn nhãn cho hình ảnh cá nhân.....	36
Figure 22. Clone Yolov5 và cài đặt các dependencies	37
Figure 23. Copy path và detect hình ảnh	37
Figure 24. Ảnh nhận diện được detect và lưu trữ trong tệp run.....	38
Figure 25. Kết quả detect đa hình ảnh	38
Figure 26. Clone và tải YOLOv5.....	39
Figure 27. Unrar coco128	39
Figure 28. Gán đường dẫn vào tệp coco128.yaml	39
Figure 29. Code tập train	39
Figure 30. Hiệu số khi được epoch 50 lần	40
Figure 31. Hiệu số khi được epoch 3 lần	40
Figure 32. Giá trị được dán nhãn trong tệp val_batch0_labels.jpg.....	41
Figure 33. Copy path vào file detect.....	41
Figure 34. Code và kết quả detect.....	41
Figure 35. Số confidence mô hình dự đoán cho các tệp ảnh	42
Figure 36. Tạo file.....	43
Figure 37. Install thư viện và YOLOv5	43
Figure 38. Code của app.py	44
Figure 39. Thiết kế giao diện	44
Figure 40. Chạy chương trình.....	44
Figure 41. Kết quả nhận diện trên website	45
Figure 42. Ảnh người dùng upload để nhận diện được gửi về hệ thống	45
Figure 43. Code của tệp detect_webcam.py	46
Figure 44. Sử dụng webcam để nhận diện.....	46

DANH MỤC BẢNG

Table 1. So sánh giữa các mô hình.....	8
Table 2. Chấm điểm dựa trên tiêu chí mô hình.....	9
Table 3. Bảng tổng hợp kết quả thí nghiệm khi train 2 dataset.....	47

TÓM TẮT ĐỒ ÁN

Tên đề tài:

Nhận diện đối tượng bằng thuật toán YOLOv5.

Hoạt động:

Nghiên cứu và áp dụng thuật toán YOLOv5 để nhận diện đối tượng từ hình ảnh hoặc video. Đồ án bao gồm việc trình bày cơ sở lý thuyết, phương pháp giải, chuẩn bị dữ liệu, huấn luyện mô hình, triển khai hệ thống và thiết kế một ứng dụng demo trực quan.

Nội dung chính:

Chương 1: Tổng quan về YOLOv5

- Giới thiệu YOLOv5, một thuật toán nhận diện vật thể nhanh và chính xác.
- Trình bày các đặc điểm nổi bật của YOLOv5, so sánh với các phiên bản YOLO trước.

Chương 2: Cơ sở lý thuyết

- Nguyên lý hoạt động của YOLOv5.
- Kiến trúc và các thành phần chính của thuật toán.

Chương 3: Chuẩn bị dữ liệu

- Thu thập và xử lý dữ liệu từ bộ dataset coco128 và dữ liệu tự tạo.
- Thực hiện gắn nhãn bằng công cụ MakeSense.AI và tổ chức dữ liệu theo cấu trúc của YOLOv5.

Chương 4: Cài đặt chương trình

- Hướng dẫn cài đặt YOLOv5 trên Google Colab.
- Quá trình huấn luyện mô hình với dữ liệu coco128 và dữ liệu cá nhân.
- Đánh giá kết quả dựa trên số lần epoch và độ chính xác (confidence).

Chương 5: Thiết kế hệ thống

- Xây dựng giao diện website và demo nhận diện bằng webcam.
- Triển khai ứng dụng với giao diện thân thiện, người dùng có thể tải lên và nhận diện đối tượng trong thời gian thực.

Chương 6: Đánh giá kết quả

- Tổng kết kết quả thực nghiệm, nhận xét và đánh giá dựa trên số liệu thực nghiệm từ đó nêu ra những ưu điểm, hạn chế và hướng phát triển cho đề tài.

CHƯƠNG I: GIỚI THIỆU ĐỀ TÀI

1.1. Lý do chọn đề tài

Ngày nay, trí tuệ nhân tạo - Artificial Intelligence (AI) đang ngày càng phổ biến và góp phần thay đổi sâu sắc nhiều khía cạnh trong cuộc sống hằng ngày. Trong đó thị giác máy tính - Computer Vision (CV) là một lĩnh vực quan trọng của AI bao gồm các phương pháp thu nhận, xử lý ảnh kỹ thuật số, phân tích và nhận dạng các hình ảnh.

Mạng Noron học sâu (Deep learning Network) là lĩnh vực nghiên cứu các thuật toán, chương trình máy tính để máy tính có thể học tập và đưa ra những dự đoán như con người. Nó được ứng dụng vào nhiều ứng dụng khác nhau như khoa học, kỹ thuật, các lĩnh vực đời sống khác cũng như các ứng dụng về phân loại và phát hiện đối tượng. Một ví dụ điển hình là CNN (Convolutional Neural Network) áp dụng để nhận dạng tự động, tìm hiểu các mẫu phân biệt từ ảnh bằng cách xếp chồng liên tiếp các lớp lên nhau và trong nhiều ứng dụng, CNN hiện nay được coi là trình phân loại ảnh mạnh và thúc đẩy các công nghệ trong lĩnh vực thị giác máy tính, làm đòn bẩy cho quá trình học máy. Nhưng bên cạnh đó, để phân loại được một đối tượng thì công nghệ CNN tiêu tốn cực lớn về tài nguyên như băng thông, bộ nhớ và khả năng xử lý của phần cứng.

Để giảm thiểu những tài nguyên tiêu hao này, những thuật toán, mô hình giải thuật theo thời gian được ra đời ngày càng nhiều và trong đó có mô hình thuật toán YOLOv5 cho bài toán nhận diện, cụ thể là ứng dụng vào bài toán “Nhận diện đối tượng với thuật toán YOLOv5”.

Nhận diện vật thể là một trong những bài toán quan trọng trong lĩnh vực thị giác máy tính, có ứng dụng rộng rãi trong thực tế, từ an ninh giám sát, giao thông thông minh đến tự động hóa và robot. Thuật toán YOLOv5 là một trong những phương pháp tiên tiến hiện nay, nổi bật với khả năng nhận diện chính xác và thời gian xử lý nhanh. Việc nghiên cứu và triển khai thuật toán này không chỉ giúp nắm vững kiến thức lý thuyết mà còn cung cấp nền tảng ứng dụng vào thực tế, đặc biệt trong bối cảnh dữ liệu lớn và yêu cầu xử lý thời gian thực ngày càng phổ biến.

1.2. Mục tiêu của đồ án

- Tìm hiểu cơ chế hoạt động và các thành phần của thuật toán YOLOv5.
- Xây dựng được một mô hình có khả năng huấn luyện các tập dữ liệu vật thể khác nhau.
- Nhận diện được tất cả các vật thể có trong tập dữ liệu.
- Đánh giá hiệu suất của thuật toán qua các chỉ số.

1.3. Giới hạn đề tài

- Trong dữ liệu coco128 thu thập được không có ảnh và nhãn gốc chỉ có 128 ảnh train và labels được cung cấp thử nghiệm, tuy nhiên mô hình đã được train sẵn nên vẫn có thể nhận diện tốt dù không có tệp ảnh, nhãn. Nhận diện với dữ liệu lớn với hơn 80 đối tượng.
- Dữ liệu tự tạo giới hạn về số lượng ảnh, có thể sử dụng roboflow để gia tăng số lượng nhưng bị hạn chế về nguồn lực.

1.4. Phương pháp nghiên cứu

- Dựa trên các kiến thức đã được học, tìm hiểu được về cách huấn luyện một mạng nơ-ron.
- Thu thập tài liệu, tham khảo những ứng dụng liên quan đã có trước đó.

1.5. Phạm vi thực hiện

- Nhóm bài toán: Nhận diện đối tượng – một nhánh cụ thể của bài toán thị giác máy tính, dữ liệu và nhãn.
- Nguồn gốc dữ liệu:
 1. Hình ảnh: Tự tạo và thu thập trên Internet.
 2. Code: Tham khảo Github và tự code.
 3. Các dữ liệu khác: Thu thập trên Internet, tự tạo.

CHƯƠNG II: CƠ SỞ LÝ THUYẾT

2.1. Tổng quan về YOLO

2.1.1. Lịch sử phát triển

YOLO lần đầu tiên được giới thiệu vào tháng 5 năm 2016 bởi Joseph Redmon trong bài báo *"You Only Look Once: Unified, Real-Time Object Detection"*. Đây là một bước tiến quan trọng trong lĩnh vực phát hiện đối tượng (Object Detection), giúp đơn giản hóa quá trình xử lý thông qua việc sử dụng một mạng duy nhất để thực hiện cả hai nhiệm vụ phát hiện và phân loại.

Vào tháng 12 năm 2017, Joseph Redmond tiếp tục giới thiệu một phiên bản cải tiến khác của YOLO thông qua bài báo *"YOLO9000: Better, Faster, Stronger"*. Phiên bản này được gọi là YOLO9000 và đã mang đến nhiều cải tiến vượt bậc trong lĩnh vực phát hiện đối tượng.

Chỉ sau chưa đầy một năm, vào tháng 4 năm 2018, một trong những phiên bản được mong đợi nhất của YOLO là YOLOv3 chính thức ra mắt. Phiên bản này được trình bày trong bài báo *"YOLOv3: An Incremental Improvement"* và đã bổ sung nhiều cải tiến về hiệu suất và khả năng phát hiện đối tượng ở nhiều quy mô khác nhau.

Hai năm sau đó, vào tháng 4 năm 2020, Alexey Bochkovsky công bố YOLOv4 qua bài báo *"YOLOv4: Optimal Speed and Accuracy of Object Detection"*. YOLOv4 mang đến những tính năng vượt trội so với YOLOv3, không chỉ tăng đáng kể tốc độ mà còn cải thiện rõ rệt độ chính xác trung bình. Phiên bản này được đánh giá là một bước tiến lớn, đáp ứng tốt các yêu cầu khắt khe của các ứng dụng phát hiện đối tượng hiện đại.

Vào tháng 6 năm 2020, Glenn Jocher phát hành YOLOv5 dựa trên PyTorch, mang lại nhiều cải tiến về hiệu suất và dễ sử dụng. Tuy nhiên, YOLOv5 chưa được công bố trong các tạp chí khoa học peer-reviewed, do đó không có bài báo chính thức về mô hình này. Nhưng, điều đặc biệt phiên bản này tuyệt vời và vượt trội hơn các đàn anh trước đó được thể hiện qua biểu đồ sau.

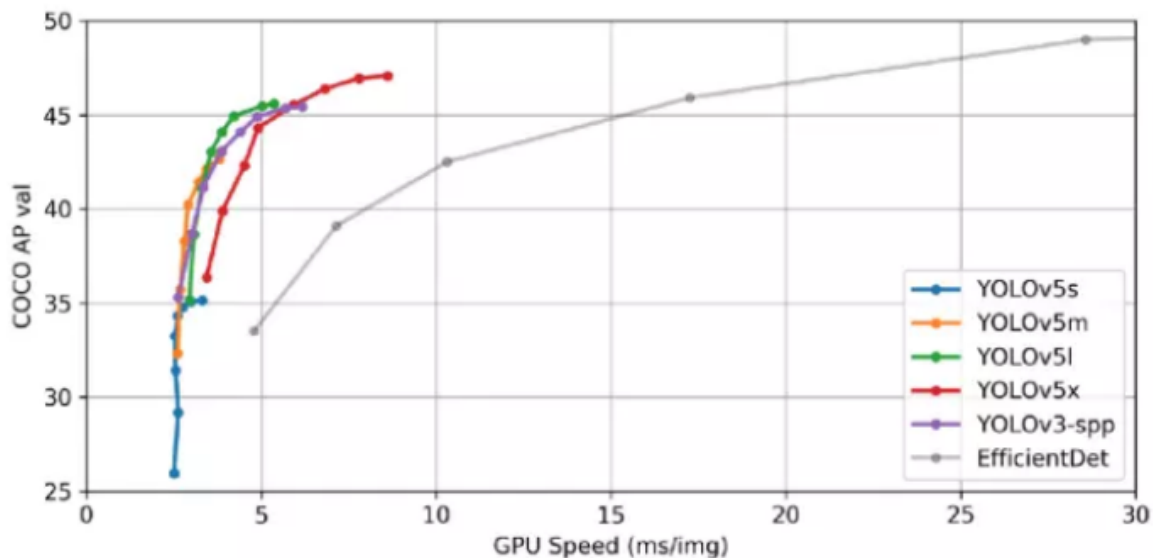


Figure 1. So sánh các phiên bản YOLOv5

(Nguồn [Responding to the Controversy about YOLOv5](#))

Mặc dù các phiên bản mới hơn như YOLOv6, YOLOv7 và YOLOv8 đã được phát triển, nhưng YOLOv5 vẫn được ưa chuộng trong nhiều ứng dụng nhờ vào sự ổn định, dễ sử dụng và hiệu suất cao. Việc lựa chọn phiên bản YOLO phù hợp phụ thuộc vào yêu cầu cụ thể của từng ứng dụng, bao gồm độ chính xác, tốc độ và khả năng triển khai trên các nền tảng khác nhau.

2.1.2. So sánh các mô hình

Tiêu chí	YOLOv4	YOLOv5	Faster R-CNN	SSD
Hiệu suất	Cao, phù hợp cho ứng dụng thời gian thực	Rất cao, tối ưu hóa cả YOLOv4	Trung bình đến thấp, không tốt cho thời gian thực	Cao, nhưng thấp hơn YOLOv4,5
Độ chính xác	Cao, nhưng có thể gặp khó với đối tượng nhỏ	Rất cao, cải thiện so với YOLOv4	Rất cao, tốt trong việc nhận diện đối tượng nhỏ	Đơn giản, dễ triển khai nhưng hiệu suất không bằng các mô hình khác
Độ phức tạp	Phức tạp, yêu cầu điều chỉnh tham số nhiều	Đơn giản hơn trong sử dụng và điều chỉnh	Rất phức tạp, yêu cầu nhiều tài nguyên và điều chỉnh	Đơn giản, dễ triển khai nhưng hiệu suất không bằng các mô hình khác.

Khả năng tổng quát hóa	Tốt, hoạt động trong nhiều bối cảnh	Rất tốt, cải thiện hơn YOLOv4	Rất tốt, nhưng yêu cầu huấn luyện kỹ lưỡng	Tốt, nhưng không bằng Faster R-CNN
Yêu cầu phần cứng	Cần GPU mạnh để đạt hiệu suất tối ưu	Tối ưu cho cả GPU và CPU, yêu cầu phần cứng thấp hơn	Cần GPU mạnh, tiêu tốn nhiều tài nguyên	Hoạt động tốt trên cả GPU và CPU, yêu cầu phần cứng trung bình
Thời gian huấn luyện	Dài hơn do mô hình phức tạp	Ngắn hơn, tối ưu để huấn luyện nhanh hơn	Dài, đòi hỏi tài nguyên lớn	Nhanh hơn Faster R-CNN, nhưng không bằng YOLOv5
Độ linh hoạt	Tốt, nhưng yêu cầu kiến thức sâu về tối ưu hóa	Cao, dễ sử dụng và điều chỉnh	Rất linh hoạt, nhưng phức tạp trong điều chỉnh	Tốt, dễ triển khai và điều chỉnh
Khả năng phát hiện đối tượng	Hạn chế trong một số trường hợp	Cải thiện hơn YOLOv4	Rất tốt, phù hợp với đối tượng nhỏ	Tốt, nhưng không bằng Faster R-CNN
Triển khai	Phức tạp hơn do yêu cầu tối ưu hóa và cấu hình	Dễ dàng hơn với công cụ và framework hỗ trợ tốt	Khó khăn hơn do độ phức tạp	Dễ dàng, ít yêu cầu tối ưu hóa phức tạp

Table 1. So sánh giữa các mô hình

Dựa trên các tiêu chí đã trình bày, bảng chấm điểm có thể được xây dựng như sau (thang điểm từ 1 đến 5, với 5 là cao nhất):

Tiêu chí	YOLOv4	YOLOv5	Faster R-CNN	SSD
Hiệu suất	4	5	2	3
Độ chính xác	4	5	5	3
Độ phức tạp	3	4	2	4
Khả năng tổng quát hóa	4	5	5	3
Yêu cầu phần cứng	3	4	2	4
Thời gian huấn luyện	3	5	2	4
Độ linh hoạt	4	5	3	4
Khả năng phát hiện đối tượng	3	5	5	3
Triển khai	3	5	2	4
Tổng điểm	31	43	28	32

Table 2. Chấm điểm dựa trên tiêu chí mô hình

Thông qua hai bảng trên, với việc nhìn nhận ưu điểm, hạn chế và chấm điểm cho các tiêu chí của thuật toán/ mô hình. Ta nhận thấy, YOLOv5 là lựa chọn phù hợp nhất cho đề tài nhận diện đối tượng.

2.1.3. Kiến trúc mạng YOLO

YOLOv5 là một phiên bản cụ thể thuộc dòng mô hình YOLO. Để hiểu rõ cách thức hoạt động của YOLOv5, cần nắm vững các nguyên lý và khái niệm cơ bản của YOLO, bao gồm kiến trúc mạng, nguyên lý vận hành và các công thức cốt lõi. Điều này đặc biệt quan trọng bởi YOLOv5 vẫn kế thừa và áp dụng nhiều công thức, hàm số từ phiên bản YOLO ban đầu. Vì vậy, phần này sẽ tập trung trình bày những nội dung nền tảng về YOLO.

Kiến trúc của YOLO bao gồm hai phần chính: Base network, được xây dựng từ các lớp convolution với mục tiêu chính là trích xuất đặc trưng, và Extra Layers, được thiết kế để phát hiện các vật thể dựa trên feature map của Base network.

Phần Base network trong YOLO chủ yếu sử dụng các lớp convolutional kết hợp với một số fully connected layer. Kiến trúc YOLO rất linh hoạt, cho phép tùy chỉnh thành nhiều phiên bản khác nhau để phù hợp với các dạng kích thước đầu vào (input shape) đa dạng.

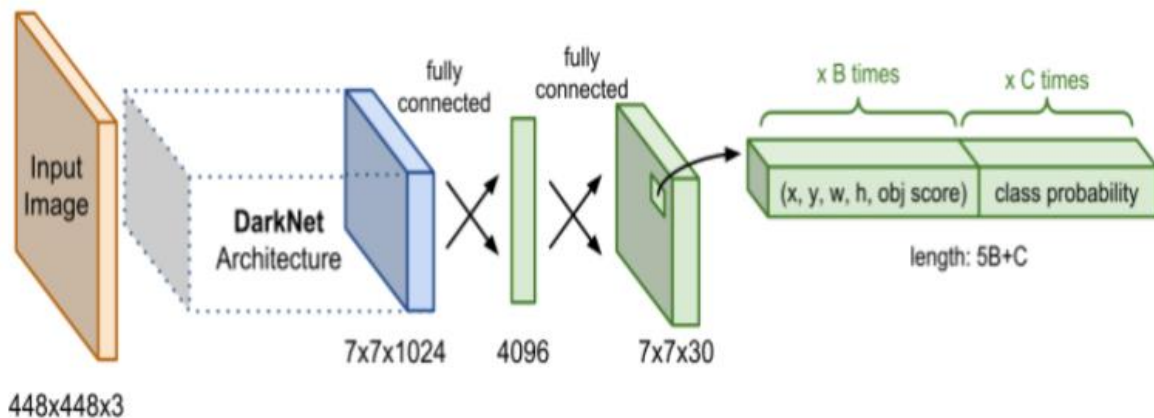


Figure 2. Kiến trúc mạng YOLO

(Nguồn: *YOLOv1 Core Ideas*)

Thành phần Darknet Architecture, thường được gọi là base network, đảm nhận vai trò trích xuất đặc trưng. Output từ base network là một feature map với kích thước 7x7x1024, được sử dụng làm đầu vào cho các Extra Layers. Những lớp này đảm nhận nhiệm vụ dự đoán nhãn và tọa độ của bounding box cho các vật thể.

Trong phiên bản thứ ba của YOLO, hay YOLOv3, tác giả đã áp dụng một mạng feature extractor mang tên Darknet-53. Kiến trúc này bao gồm 53 lớp convolution được liên kết liên tiếp, trong đó mỗi lớp đi kèm với batch normalization và activation function Leaky ReLU. Để giảm kích thước đầu ra sau mỗi lớp convolution, tác giả đã sử dụng các bộ lọc với kích thước kernel bằng 2 để thực hiện downsampling. Phương pháp này giúp giảm thiểu số lượng tham số trong mô hình, từ đó tăng hiệu quả tính toán.

	Type	Filters	Size	Output
	Convolutional	32	3×3	256×256
	Convolutional	64	$3 \times 3 / 2$	128×128
1x	Convolutional	32	1×1	
	Convolutional	64	3×3	
	Residual			128×128
	Convolutional	128	$3 \times 3 / 2$	64×64
2x	Convolutional	64	1×1	
	Convolutional	128	3×3	
	Residual			64×64
	Convolutional	256	$3 \times 3 / 2$	32×32
8x	Convolutional	128	1×1	
	Convolutional	256	3×3	
	Residual			32×32
	Convolutional	512	$3 \times 3 / 2$	16×16
8x	Convolutional	256	1×1	
	Convolutional	512	3×3	
	Residual			16×16
	Convolutional	1024	$3 \times 3 / 2$	8×8
4x	Convolutional	512	1×1	
	Convolutional	1024	3×3	
	Residual			8×8
	Avgpool		Global	
	Connected		1000	
	Softmax			

Figure 3. Các layer trong mạng darknet-53

(Nguồn: Tổng hợp kiến thức từ YOLOv1 đến YOLOv5 (Phần 2))

Khi các bức ảnh được đưa vào mô hình, chúng sẽ được scale về cùng một kích thước, đảm bảo phù hợp với input shape của mô hình trước khi được gom thành các batch để tiến hành huấn luyện.

Hiện tại, YOLO hỗ trợ hai kích thước đầu vào chính là 416×416 và 608×608 . Mỗi loại đầu vào sẽ có một thiết kế layer riêng biệt, tối ưu hóa cho kích thước tương ứng. Khi ảnh đi qua các lớp convolutional, kích thước shape sẽ giảm dần theo cấp số nhân là 2. Kết quả cuối cùng là một feature map có kích thước nhỏ gọn, giúp dự đoán các vật thể trên từng ô của feature map.

Kích thước của feature map sẽ phụ thuộc vào kích thước đầu vào. Với input 416×416 , các feature map được tạo ra có kích thước lần lượt là 13×13 , 26×26 , và 52×52 . Đối với input 608×608 , các feature map sẽ có kích thước tương ứng là 19×19 , 38×38 , và 72×72 .

2.1.4. Nguyên lý hoạt động của mạng YOLO

Đầu vào của mô hình là một hình ảnh, và nhiệm vụ của mô hình là xác định xem trong hình ảnh đó có đối tượng nào không. Nếu có, mô hình sẽ xác định tọa độ của các đối tượng trong ảnh.

Ảnh đầu vào được chia thành lưới có kích thước $S \times S$, với các giá trị thường gặp như 3×3 , 7×7 , hoặc 9×9 ... Quá trình chia lưới này đóng vai trò quan trọng trong việc phát hiện đối tượng của mô hình, vì số lượng và kích thước các ô lưới có thể ảnh hưởng trực tiếp đến độ chính xác trong việc định vị và phân loại các đối tượng.

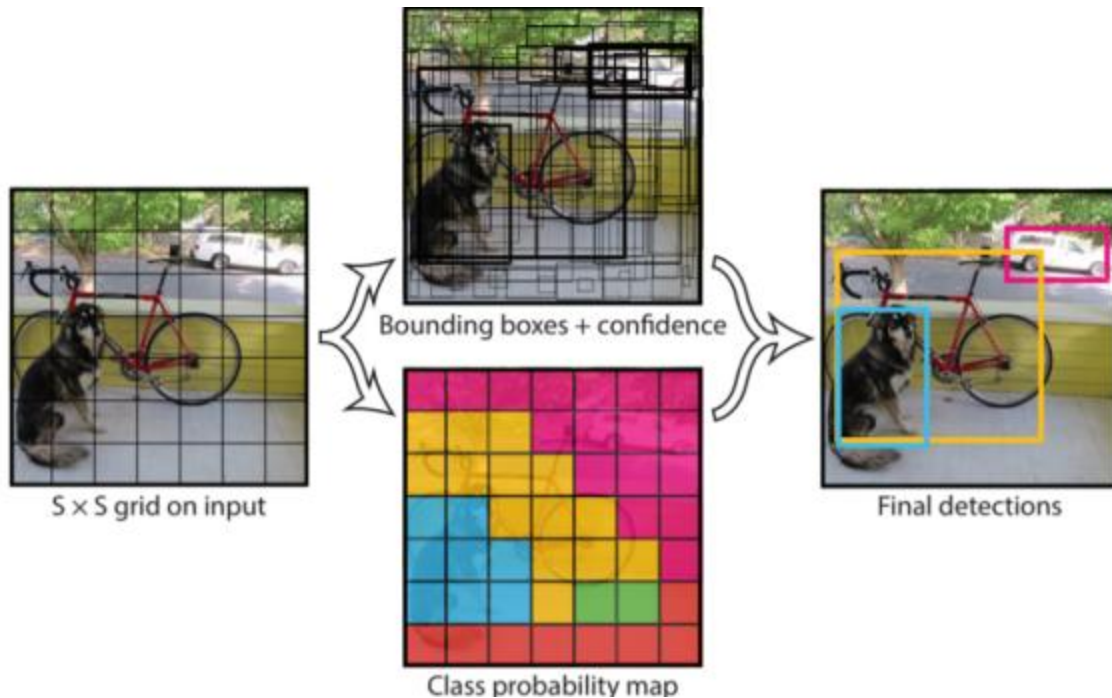


Figure 4. Cách hoạt động của mạng YOLO

(Nguồn: *YOLO: Real-Time Object Detection*)

Với đầu vào là một hình ảnh, đầu ra của mô hình là một ma trận ba chiều có kích thước $S \times S \times (5 \times N + M)$, trong đó mỗi ô chứa $(5 \times N + M)$ tham số. Ở đây:

- N là số lượng bounding box cần dự đoán trong mỗi ô.
- M là số lượng class mà mô hình có thể nhận diện.

Lấy ví dụ hình ảnh được chia thành lưới 7×7 ô, mỗi ô cần dự đoán 2 bounding box và nhận diện 3 loại đối tượng (ví dụ: chó, ô tô, xe đạp). Trong trường hợp này:

- Mỗi ô sẽ có 13 tham số ($5 \times 2 + 3$).
- Output của mô hình sẽ là một ma trận kích thước $7 \times 7 \times 13$, cung cấp thông tin cho tổng cộng 98 bounding box ($7 \times 7 \times 2$).

2.2. Output của YOLO

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

- Công thức:

$$\underline{y}^T = [\underbrace{p_0, \langle t_x, t_y, t_w, t_h \rangle}_{\text{bounding box}}, \underbrace{\langle \underline{p_1}, \underline{p_2}, \dots, \underline{p_c} \rangle}_{\text{score of } c \text{ classes}}]$$

Trong đó:

- p_0 : là xác suất dự báo vật thể xuất hiện trong bounding box.
- $\langle t_x, t_y, t_w, t_h \rangle$: giúp xác định bounding box. Trong đó t_x, t_y là tọa độ tâm và t_w, t_h là kích thước rộng, dài của bounding box.
- p_1, p_2, \dots, p_c : là véc tơ phân phối xác suất dự báo của các classes.

Việc hiểu rõ cấu trúc đầu ra (output) của mô hình là rất quan trọng để cấu hình các tham số một cách chính xác khi huấn luyện mô hình trên các nền tảng mã nguồn mở như Darknet. Đầu ra của mô hình được xác định dựa trên số lượng classes theo công thức $(n_class + 5)$. Chẳng hạn, nếu mô hình được huấn luyện với 80 classes, số lượng tham số output sẽ là 85.

Nếu áp dụng 3 anchors/cell, số lượng tham số output sẽ được tính toán dựa trên tổng các bounding boxes từ các anchors này:

$$(n_class + 5) \times 3 = 85 \times 3 = 255$$

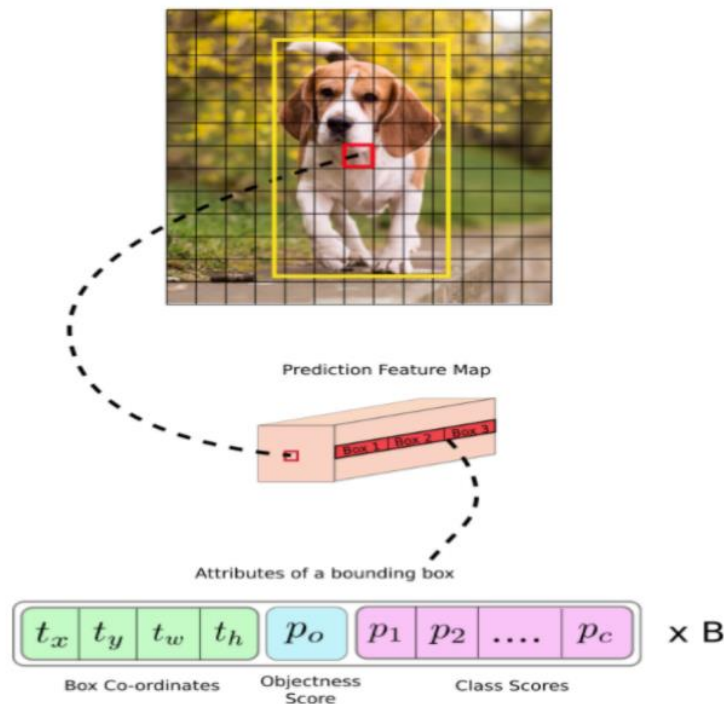


Figure 5. Kiến trúc một output của model YOLO

(Nguồn: [Drone detection using YOLOv3 with transfer learning on NVIDIA Jetson TX2](#) | Semantic Scholar)

Xét ví dụ hình ảnh đầu vào có feature map với kích thước 13×13 . Trên mỗi cell của feature map, chúng ta chọn ra 3 anchor boxes có kích thước khác nhau (Box 1, Box 2, Box 3), với điều kiện tâm của các anchor boxes trùng với vị trí cell. Kết quả đầu ra của YOLO sẽ là một vector kết hợp (concatenate) từ 3 bounding boxes.

Mỗi bounding box bao gồm các thuộc tính (attributes) cụ thể như tọa độ, kích thước, và xác suất chứa các class. Các thuộc tính này được sắp xếp và mô tả chi tiết trong các dòng cuối cùng của hình minh họa.

2.2.1. Dự báo trên nhiều feature map

Tương tự như SSD, YOLO (đặc biệt là YOLOv3) thực hiện dự báo trên nhiều feature map. Các feature map ban đầu với kích thước nhỏ được thiết kế để dự đoán các vật thể có kích thước lớn. Trong khi đó, các feature map ở những tầng sau có kích thước lớn hơn, kết hợp với việc giữ nguyên kích thước của anchor boxes, sẽ hỗ trợ dự báo hiệu quả các vật thể có kích thước nhỏ.

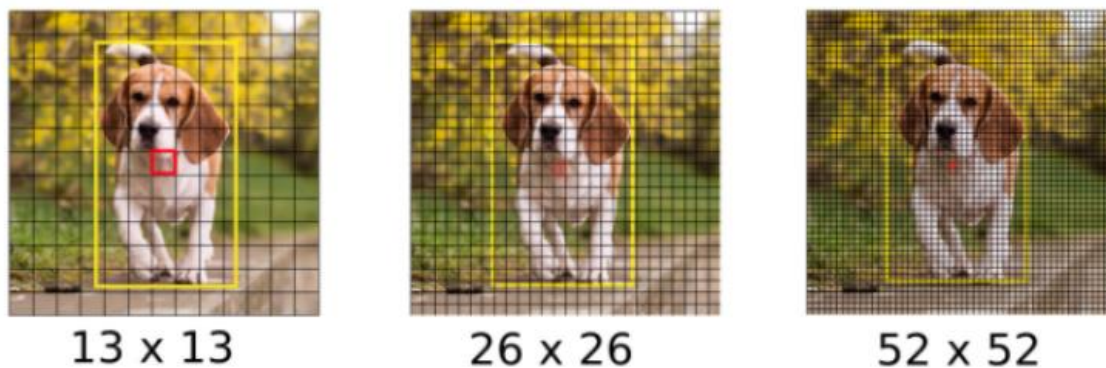


Figure 6. Các feature maps của mạng YOLOv3 với input shape là 416×416 , output là 3 feature maps có kích thước lần lượt là 13×13 , 26×26 và 52×52 .

(Nguồn: [Drone detection using YOLOv3 with transfer learning on NVIDIA Jetson TX2 | Semantic Scholar](#))

Trên mỗi một cell của các feature map chúng ta sẽ áp dụng 3 anchor box để dự đoán vật thể. Như vậy số lượng các anchor box khác nhau trong một mô hình YOLO sẽ là 9 (3 feature map x 3 anchor box).

Đồng thời trên một feature map hình vuông $S \times S$, mô hình YOLOv3 sinh ra một số lượng anchor box là: $S \times S \times 3$. Như vậy số lượng anchor boxes trên một bức ảnh sẽ là:

$$(13 \times 13 + 26 + 52 \times 52) \times 3 = 10647 \text{ (anchor box)}$$

Đây là một số lượng rất lớn và là nguyên nhân khiến quá trình huấn luyện mô hình YOLO vô cùng chậm bởi chúng ta cần dự báo đồng thời nhãn và bounding box trên đồng thời 10647 bounding boxes.

Một số lưu ý khi huấn luyện YOLO:

- Yêu cầu về RAM: Khi huấn luyện YOLO, mô hình cần một dung lượng RAM lớn để lưu trữ số lượng lớn bounding boxes (ví dụ: 10,647 bounding boxes như trong kiến trúc mặc định).
- Kích thước batch: Không thể thiết lập giá trị batch_size quá lớn như trong các mô hình classification, vì điều này dễ dẫn đến tình trạng Out of Memory (OOM). Để khắc phục, thư viện Darknet được sử dụng trong YOLO chia nhỏ một batch thành các subdivisions sao cho phù hợp với dung lượng RAM khả dụng.
- Thời gian xử lý: Thời gian xử lý của một step trên YOLO thường lâu hơn rất nhiều so với các mô hình classification. Do đó, việc giới hạn số lượng steps khi huấn luyện YOLO là điều cần thiết.
 1. Với các tác vụ nhận diện ít hơn 5 classes, thiết lập số lượng steps dưới 5,000 có thể cho ra kết quả chấp nhận được.
 2. Với các mô hình có nhiều classes hơn, số lượng steps cần được tăng dần theo cấp số nhân, tùy thuộc vào mục tiêu và dữ liệu của người dùng.

2.2.2. Anchor box

Để dự đoán bounding box cho các vật thể, YOLO sử dụng anchor boxes làm cơ sở ước lượng. Các anchor boxes được xác định trước và có nhiệm vụ bao quanh các vật thể trong hình ảnh một cách tương đối chính xác. Sau đó, thuật toán regression bounding box sẽ tinh chỉnh lại các anchor boxes để tạo ra bounding boxes dự đoán, phù hợp hơn với vật thể. Trong một mô hình YOLO:

- Phân phối vật thể: Mỗi vật thể trong ảnh huấn luyện sẽ được gán vào một anchor box.
- Xử lý trùng lặp:
 1. Trong trường hợp có nhiều hơn một anchor box (từ hai anchor boxes trở lên) cùng bao quanh một vật thể, thuật toán sẽ chọn anchor box có IoU (Intersection over Union) cao nhất với ground truth bounding box.
 2. Việc này đảm bảo rằng mỗi vật thể được ánh xạ vào anchor box phù hợp nhất để tối ưu hiệu suất dự đoán của mô hình.

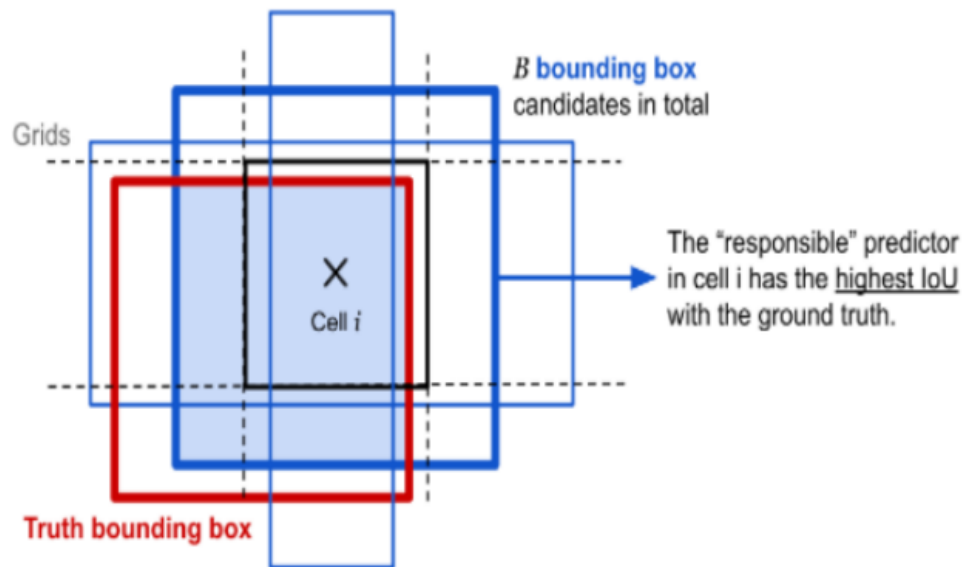


Figure 7. Xác định anchor box cho một vật thể

(Nguồn: Bài 25 - YOLO You Only Look Once)

Từ Cell i ta xác định được 3 anchor boxes viền xanh như trong hình. Cả 3 anchor boxes này đều giao nhau với bounding box của vật thể. Tuy nhiên chỉ anchor box có đường viền dày nhất màu xanh được lựa chọn làm anchor box cho vật thể bởi nó có IoU so với ground truth bounding box là cao nhất.

Mỗi một vật thể trong hình ảnh huấn luyện được phân bổ về một cell trên feature map mà chứa điểm midpoint của vật thể. Chẳng hạn như hình chú chó trong hình 3 sẽ được phân về cho cell màu đỏ vì điểm midpoint của ảnh chú chó rơi vào đúng cell này. Từ cell ta sẽ xác định các anchor boxes bao quanh hình ảnh chú chó.

Như vậy khi xác định một vật thể ta sẽ cần xác định 2 thành phần gắn liền với nó là (cell, anchor box). Không chỉ riêng mình cell hoặc chỉ mình anchor box.

Một số trường hợp 2 vật thể bị trùng midpoint, mặc dù rất hiếm khi xảy ra, thuật toán sẽ rất khó xác định được class cho chúng.

Anchor box example

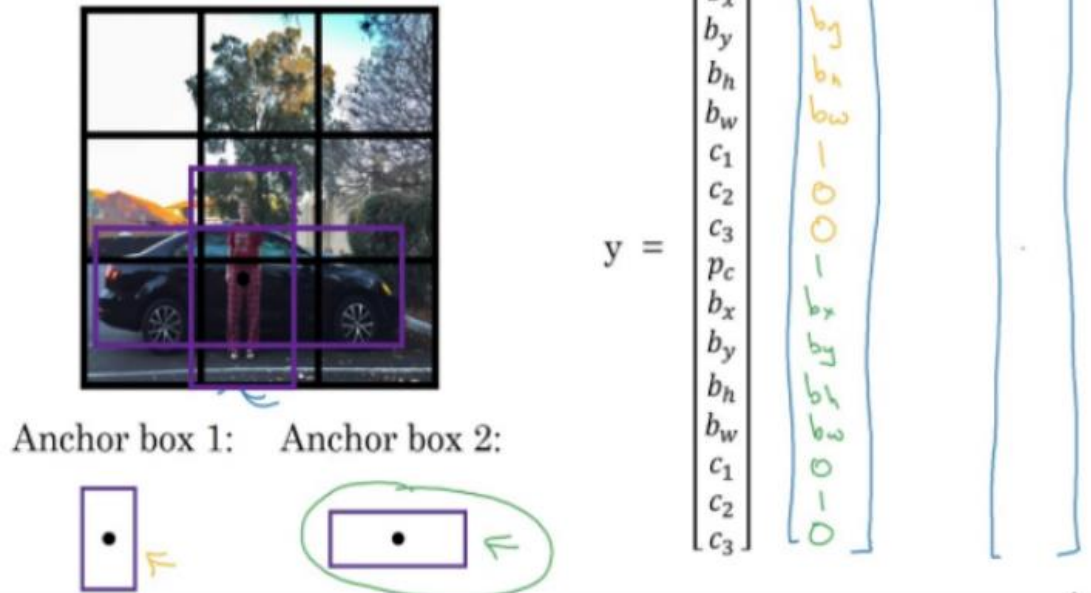


Figure 8. Khi 2 vật thể người và xe trùng mid point và cùng thuộc một cell. Thuật toán sẽ cần thêm những lượt tiebreak để quyết định đâu là class cho cell.

(Nguồn: [YOLO Architecture. YOLO \(You Only Look Once \) algorithm...](#) | by Divya | Medium)

2.2.3 Hàm mất mát (Loss Function)

Sau khi đã định nghĩa được những thông tin mà mô hình cần phải dự đoán, và kiến trúc của mô hình CNN. Bây giờ là lúc mà chúng ta sẽ định nghĩa hàm lỗi.

YOLO sử dụng hàm độ lỗi bình phương giữ dự đoán và nhãn để tính độ lỗi cho mô hình. Cụ thể, độ lỗi tổng của chúng ta sẽ là tổng của 3 độ lỗi con sau:

- Độ lỗi của việc dự đoán loại nhãn của Object-Classification loss.
- Độ lỗi của dự đoán tạo độ cũng như chiều dài, rộng của boundary box - Localization loss.
- Độ lỗi của ô vuông có chứa object nào hay không - Confidence loss.

Chúng ta mong muốn hàm lỗi có chức năng sau:

- Trong quá trình huấn luyện, mô hình sẽ nhìn vào những ô vuông có chứa object. Tăng classification score lớp đúng của object đó lên.
- Sau đó, cũng nhìn vào ô vuông đó, tìm boundary box tốt nhất trong 2 boxes được dự đoán.

Tăng localization score của boundary box đó lên, thay đổi thông tin boundary box để gần đúng với nhãn. Đối với những ô vuông không chứa object, giảm confidence score và chúng ta sẽ không quan tâm đến classification score và localization score của những ô vuông này.

2.2.3.1. Classification Loss

Chúng ta chỉ tính classification loss cho những ô vuông được đánh nhãn là có object. Classification loss tại những ô vuông đó được tính bằng đồ lỗi bình phương giữa nhãn được dự đoán và nhãn đúng của nó.

- Công thức:

$$\underline{L_{\text{classification}}} = \sum_{i=0}^{s^2} \Pi_i^{obj} \sum_{c \in \text{class}} (p_i(c) - \hat{p}_i(c))^2$$

Trong đó:

- Π_i^{obj} : bằng 1 nếu ô vuông đang xét có object, ngược lại bằng 0.
- $p_i(c)$: là xác suất có điều kiện của lớp c tại ô vuông tương ứng mà mô hình dự đoán.

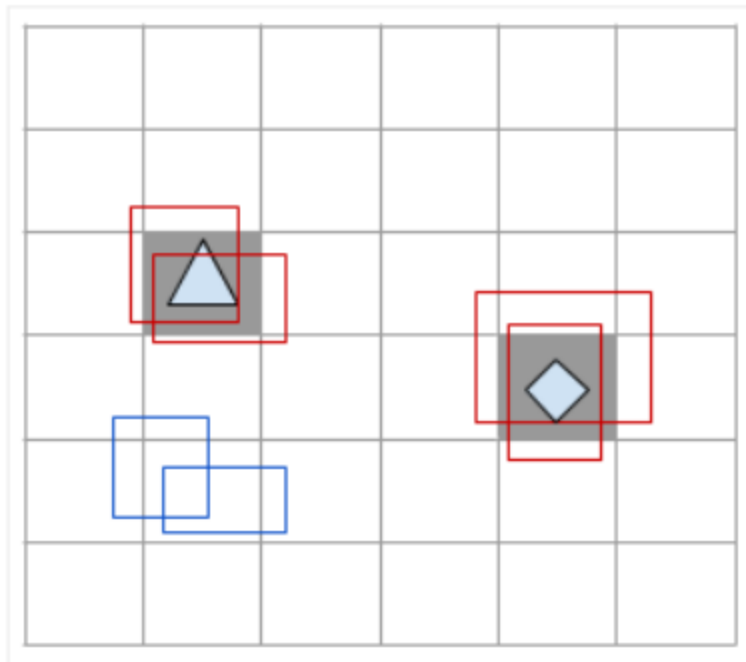


Figure 9. Tính toán Loss Function cho 2 object: tam giác và hình thoi.

(Nguồn: Bài 25 - YOLO You Only Look Once)

Ví dụ: trong hình minh họa ở trên, chúng ta có 2 object tại ô vuông (dòng,cột) là (2,1) và (3,4), chứa object là hình tam giác và hình tứ giác đều. Độ lỗi classification loss chỉ tính cho 2 object này mà ko quan tâm đến những ô vuông khác. Lúc cài đặt chúng ta cần lưu ý phải nhân với một mask để triệt tiêu giá trị lỗi tại những ô vuông ko quan tâm.

2.2.3.2. Localization Loss

Localization loss dùng để tính giá trị lỗi cho boundary box được dự đoán bao gồm offset x,y và chiều dài, rộng so với nhãn chính xác của chúng ta. Các bạn nên lưu ý rằng, chúng ta không tính toán trực tiếp giá trị lỗi này trên kích thước của ảnh mà cần chuẩn dưới kính thước ảnh về đoạn [0-1] đối với tọa độ điểm tâm, và không dự đoán trực tiếp điểm tâm mà phải dự đoán giá trị lệch offset x,y so với ô vuông tương ứng. Việc chuẩn hóa kích thước ảnh và dự đoán offset làm cho mô hình nhanh hội tụ hơn so với việc dự đoán giá trị mặc định.

$$L_{localization} = \sum_{i=0}^{S^2} \sum_{j=0}^B \prod_{ij}^{obj} [(offsetx_i - \widehat{offsetx}_i)^2 + (offsety_i - \widehat{offsety}_i)^2 + (width_i - \widehat{width}_i)^2 + (height_i - \widehat{height}_i)^2]$$

Độ lỗi localization loss được tính bằng tổng độ lỗi bình phương của offsetx, offsety và chiều dài, rộng trên tất cả các ô vuông có chứa object. Tại mỗi ô vuông đúng, ta chọn 1 boundary box có IoU (Intersect over union) tốt nhất, rồi sau đó tính độ lỗi theo các boundary box này. Theo hình minh họa trên chúng ta có 4 boundary box tại ô vuông đúng có viền màu đỏ, chúng ta chọn 1 box tại mỗi ô vuông để tính độ lỗi. Còn box xanh được bỏ qua.

Localization loss là độ lỗi quan trọng nhất trong 3 loại độ lỗi trên. Do đó, ta cần đặt trọng số cao hơn cho độ lỗi này.

2.2.3.3 Confidence Loss

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

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

Độ lỗi này là độ lỗi bình phương của dự đoán boundary đó chứa object với nhãn thực tế của ô vuông tại vị trí tương ứng, chúng ta lưu ý rằng, độ lỗi tại ô vuông mà nhãn chứa object quan trọng hơn là độ lỗi tại ô vuông không chứa object, do đó chúng ta cần sử dụng hệ số lamda để cân bằng điều này.

Tổng kết lại, tổng lỗi của chúng ta sẽ bằng tổng của 3 loại độ lỗi trên:

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

2.3. Dự báo bounding box

Để dự báo bounding box cho một vật thể chúng ta dựa trên một phép biến đổi từ anchor box và cell.

YOLOv2 và YOLOv3 dự đoán bounding box sao cho nó sẽ không lệch khỏi vị trí trung tâm quá nhiều. Nếu bounding box dự đoán có thể đặt vào bất kỳ phần nào của hình ảnh, như trong mạng regional proposal network, việc huấn luyện mô hình có thể trở nên không ổn định.

Cho một anchor box có kích thước (p_w, p_h) tại cell nằm trên feature map 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 cell và 2 tham số sau là tỷ lệ so với anchor box. Và các tham số này sẽ giúp xác định bounding box dự đoán b có tâm (b_x, b_y) à kích thước (b_w, b_h) thông qua hàm sigmoid và hàm exponential như các công thức bên dưới:

$$b_x = s(t_x) + c_x$$

$$b_y = s(t_y) + c_y$$

$$b_w = r_w$$

$$b_h = r_h$$

Ngoài ra do các tọa độ đã được hiệu chỉnh theo width và height của bức ảnh nên luôn có giá trị nằm trong ngưỡng $[0, 1]$. Do đó khi áp dụng hàm sigmoid giúp ta giới hạn được tọa độ không vượt quá xa các ngưỡng này.

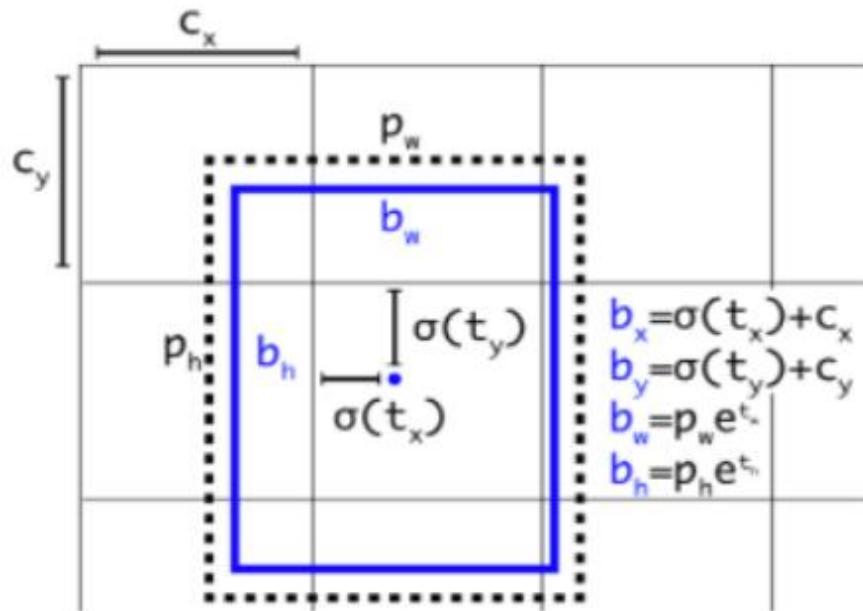


Figure 10. Công thức ước lượng bounding box từ anchor box

(Nguồn: Bài 25 - YOLO You Only Look Once)

Hình chữ nhật nét đứt bên ngoài là anchor box có kích thước là (p_w, p_h) . Tọa độ của một bounding box sẽ được xác định dựa trên đồng thời cả anchor box và cell mà nó thuộc về. Điều này giúp kiểm soát vị trí của bounding box dự đoán đâu đó quanh vị trí của cell và bounding box mà không vượt quá xa ra bên ngoài giới hạn này. Do đó quá trình huấn luyện sẽ ổn định hơn rất nhiều so với YOLOv1.

2.3.1. Non-max suppression

Do thuật toán YOLO dự báo ra rất nhiều bounding box trên một bức ảnh nên đối với những cell có vị trí gần nhau, khả năng các khung hình bị overlap 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.



Figure 11. Non-max suppression. Từ 3 bounding box ban đầu cùng bao quanh chiếc xe đã giảm xuống còn một bounding box cuối cùng.

(Nguồn: Bài 25 - YOLO You Only Look Once)

Các bước của non-max suppression:

- Bước 1: Đầu tiên chúng ta sẽ tìm cách giảm bớt số lượng các bounding box bằng cách lọc bỏ toàn bộ những bounding box có xác suất chứa vật thể nhỏ hơn một ngưỡng threshold nào đó, thường là 0.5.
- Bước 2: Đối với các bounding box giao nhau, non-max suppression sẽ lựa chọn ra một bounding box có xác suất chứa vật thể là lớn nhất. Sau đó tính toán chỉ số giao thoa IoU với các bounding box còn lại.

Nếu chỉ số này lớn hơn ngưỡng threshold thì điều đó chứng tỏ 2 bounding boxes đang overlap nhau rất cao. Ta sẽ xóa các bounding box có xác suất thấp hơn và giữ lại bounding box có xác suất cao nhất. Cuối cùng, ta thu được một bounding box duy nhất cho một vật thể.

2.4. Chi tiết về thuật toán YOLOv5

2.4.1. Khái niệm

YOLOv5 là bản cải tiến mang tính mở rộng theo một cách tự nhiên của YOLOv3 PyTorch bởi tác giả Glenn Jocher. Kho lưu trữ YOLOv3 PyTorch là điểm đến phổ biến cho các nhà phát triển để chuyển các trọng số YOLOv3 Darknet sang PyTorch và sau đó chuyển sang sản xuất.

Thuật toán YOLOv5 về cơ bản cũng thừa hưởng từ các phương pháp cơ bản của YOLO tuy nhiên YOLOv5 áp dụng một số thuật toán phát hiện vật thể nhanh tối ưu hóa các phép toán thực hiện giúp tăng tốc độ nhận diện và giảm thời gian huấn luyện một cách tối ưu.

2.4.2. Kiến trúc thuật toán YOLOv5

YOLOv5 đã cho thấy đây là phiên bản có thể cải thiện độ chính xác, tốc độ và hiệu suất so với YOLOv4. Thêm vào đó YOLOv5 còn giảm thiểu kích thước của mô hình để có thể chạy trên các thiết bị tầm trung và điện thoại thông minh.

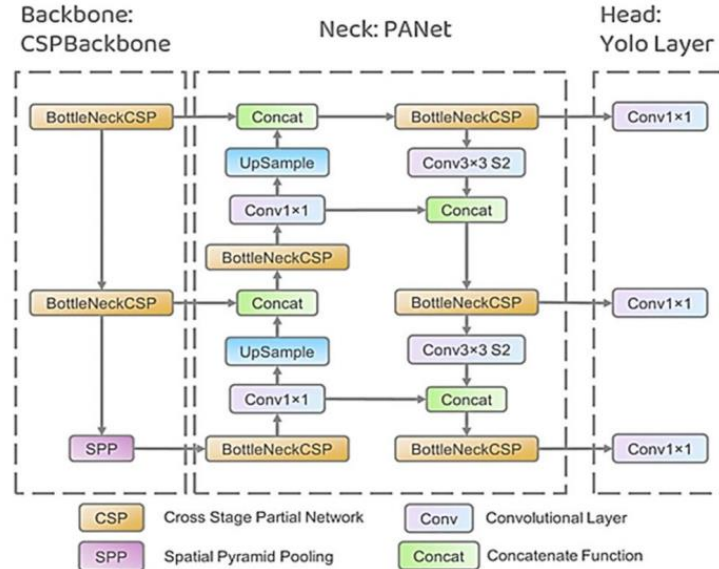


Figure 12. Kiến trúc của YOLOv5

(Nguồn: [The neural network architecture of YOLOv5. | Download Scientific Diagram](#))

Kiến trúc của YOLOv5 bao gồm ba phần chính: Backbone, Neck, và Head.

1. **Backbone:** Dựa trên kiến trúc CSPNET (Cross-Stage Partial networks). Kiến trúc CSP được dùng để xây dựng mô hình lưới mới, giúp cải thiện tốc độ và chính xác của mô hình so với các phiên bản tiền nhiệm. Kiến trúc CSP chia mạng CNN thành các tầng (stage) riêng biệt, với mỗi tầng có một phần tiền xử lý (preprocessing) và một phần hậu xử lý (postprocessing). Việc tách CNN thành các tầng nhỏ hơn giúp làm giảm độ phức tạp tính toán và giúp huấn luyện mô hình nhanh hơn.
2. **Neck:** Sử dụng mô hình trích xuất đặc trưng hiệu quả để giảm độ phức tạp và tăng tốc độ xử lý. Mô hình này được xây dựng dựa trên CSPNet với các lớp phân tích và tích hợp đặc trưng, giúp trích xuất các đặc trưng của ảnh một cách chính xác và hiệu quả.
3. **Head:** Được xây dựng trên kiến trúc của YOLOv3 nhưng được cải tiến và tối ưu tốt hơn để đạt được độ chính xác và tốc độ xử lý cao hơn.

2.4.3. Quá trình hoạt động

Đầu tiên, **Backbone** (CSPDarknet53) trích xuất các đặc trưng từ ảnh đầu vào thông qua các lớp tích chập và khối dư, giúp tối ưu hóa hiệu suất xử lý. Tiếp theo, **Neck** (kết hợp FPN và PAN) tạo ra các đặc trưng ở nhiều cấp độ, hỗ trợ nhận diện đối tượng với kích thước đa dạng. Cuối cùng, **Head** dự đoán bounding boxes, xác suất tồn tại đối tượng và nhãn phân loại thông qua các

lớp tích chập, đồng thời sử dụng kỹ thuật Anchor Boxes và Non-Maximum Suppression (NMS) để loại bỏ các bounding boxes trùng lặp.

Trong quá trình huấn luyện, mô hình tối ưu hóa bằng các hàm loss như IoU Loss, Objectness Loss, và Classification Loss, đảm bảo độ chính xác cao khi phát hiện đối tượng.

1. Đầu vào:

- YOLOv5 nhận một hình ảnh đầu vào và chuyển nó thành một tensor
- Kích thước của hình ảnh đầu vào có thể được điều chỉnh, nhưng thường được chuẩn hóa thành một kích thước cố định.

2. Đầu ra: là một tensor 3 chiều có kích thước $S \times S \times (5N + C)$, trong đó

- S là số ô dưới (grill cells) trên hình ảnh.
- N là số lượng bounding box dự đoán ở mỗi ô của lưới.
- C là số lượng lớp (class) mà mỗi ô cần dự đoán.

Mỗi ô lưới dự đoán N bounding box, mỗi bounding box bao gồm 5 thông số: tọa độ x, y , chiều rộng, chiều cao và độ tin cậy (confidence) của việc dự đoán. Ngoài ra, mỗi ô còn dự đoán xác suất cho C lớp khác nhau.

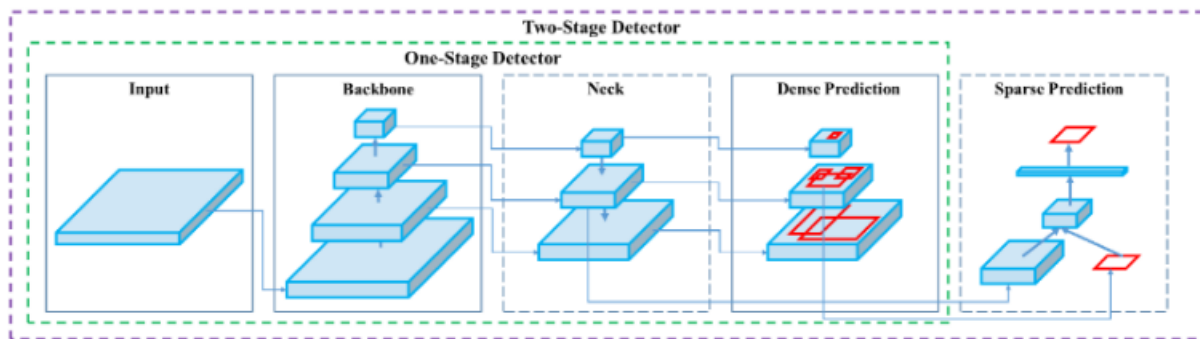


Figure 13. Quá trình nhận diện đối tượng của YOLOv5

(Nguồn: [Computer Vision Applications in Self-Driving Cars](#))

2.4.4. Phân loại YOLOv5

Khác với các bản YOLO trước, YOLOv5 cung cấp 4 phiên bản với các kiến trúc mạng khác nhau:

- YOLOv5s phiên bản nhỏ: phiên bản nhỏ gọn, tốc độ xử lý nhanh, phù hợp cho các ứng dụng yêu cầu thời gian thực và tài nguyên hạn chế.
- YOLOv5m phiên bản trung bình: cân bằng giữa tốc độ và độ chính xác, phù hợp cho các ứng dụng yêu cầu cả hai yếu tố này.
- YOLOv5l phiên bản lớn: phiên bản lớn hơn, độ chính xác cao hơn, thích hợp cho các bài toán phức tạp hơn nhưng vẫn giữ được tốc độ xử lý khá nhanh.
- YOLOv5x phiên bản cực lớn: phiên bản lớn nhất, độ chính xác cao nhất, phù hợp cho các ứng dụng yêu cầu độ chính xác tối đa dù phải đánh đổi bằng tài nguyên và thời gian xử lý.

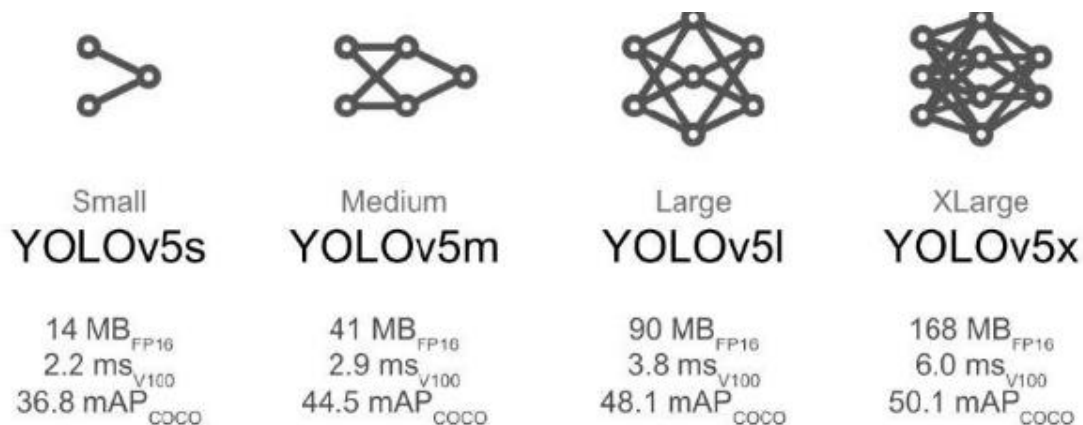


Figure 14. Các phiên bản của YOLOv5

(Nguồn: [FP16 refers for the half floating-point precision, V100 is the...](#) | Download Scientific Diagram)

Trong đề tài này, phiên bản sử dụng là YOLOv5s bởi các lý do sau:

- Tốc độ cao: YOLOv5s được thiết kế để có tốc độ xử lý nhanh, phù hợp với các ứng dụng thời gian thực.
- Hiệu quả: Dù nhỏ gọn, YOLOv5s vẫn duy trì độ chính xác đủ cao cho các tác vụ nhận diện đối tượng.
- Tài nguyên: YOLOv5s yêu cầu ít tài nguyên phần cứng hơn so với các phiên bản lớn hơn, giúp dễ dàng triển khai trên các thiết bị có tài nguyên hạn chế.

Model	AP _{val}	AP _{test}	AP ₅₀	Speed _{GPU}	FPS _{GPU}	params	FLOPS
YOLOv5s	36.6	36.6	55.8	2.1ms	476	7.5M	13.2B
YOLOv5m	43.4	43.4	62.4	3.0ms	333	21.8M	39.4B
YOLOv5l	46.6	46.7	65.4	3.9ms	256	47.8M	88.1B
YOLOv5x	48.4	48.4	66.9	6.1ms	164	89.0M	166.4B
YOLOv3-SPP	45.6	45.5	65.2	4.5ms	222	63.0M	118.0B

Figure 15. So sánh tốc độ xử lý và độ chính xác giữa các phiên bản

(Nguồn: [FP16 refers for the half floating-point precision, V100 is the...](#) | Download Scientific Diagram)

2.4.5. Nguyên lý hoạt động của YOLOv5

2.4.5.1. Các bước tính bounding box và detect trong thuật toán YOLOv5

Bước 1: Resize và chuẩn hóa ảnh

Trước tiên, chúng ta cần resize ảnh về kích thước chuẩn mà YOLOv5 yêu cầu. Điều này giúp cho mô hình có thể xử lý tất cả các ảnh đầu vào với kích thước đồng nhất. Trường hợp không cần resize khi ảnh đã đáp ứng đủ kích thước của mô hình - 640x640 pixel.

- Công thức tính tỷ lệ scale (s):

$$s = \min \left(\frac{\text{chiều rộng ảnh chuẩn}}{\text{chiều rộng ảnh gốc}}, \frac{\text{chiều cao ảnh chuẩn}}{\text{chiều cao ảnh gốc}} \right)$$

- Tính kích thước ảnh sau khi resize

$$h_{\text{new}} = \text{chiều cao gốc} \times s, \quad w_{\text{new}} = \text{chiều rộng gốc} \times s$$

Bước 2: Chia ảnh thành lưới (Grid)

Mô hình YOLOv5 chia ảnh thành các ô nhỏ (grid cells), ;ười sẽ giúp mô hình xác định các bounding box cho từng đối tượng trong ảnh. Mỗi ô sẽ chịu trách nhiệm phát hiện các đối tượng trong vùng của nó.

- Tính kích thước mỗi cell:

$$\text{Cell size} = \frac{\text{chiều dài cạnh ảnh chuẩn}}{\text{số lượng cell trên mỗi cạnh}}$$

Sau khi xác định tọa độ tâm cell chứa đối tượng, tìm chỉ số Cx, Cy.

Bước 3: Tính tọa độ Bounding Box (BBBox)

Mỗi cell sẽ dự đoán một bounding box cho đối tượng trong vùng của nó. Các thông số quan trọng để dự đoán bounding box bao gồm:

- Offset (t_x, t_y): Điều chỉnh tâm của bounding box so với tâm của cell.
- Width/Height (t_w, t_h): Điều chỉnh chiều rộng và chiều cao của bounding box.

1. Tâm bounding box (b_x, b_y)

- Công thức

$$b_x = \sigma(t_x) + c_x, \quad b_y = \sigma(t_y) + c_y$$

- Với hàm sigmoid ($\sigma(x)$) giúp đưa các giá trị về phạm vi [0,1].

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

2. Chiều rộng và chiều bounding box (b_w, b_h):

Chiều rộng và chiều cao được tính dựa trên anchor box và offset, ta có công thức sau:

$$b_w = a_w \cdot e^{t_w}, \quad b_h = a_h \cdot e^{t_h}$$

Trong đó A_w và A_h là các kích thước của các anchor box, còn T_w và T_h là các offset đã được dự đoán.

Bước 4: Quy đổi tọa độ về ảnh gốc

Sau khi tính toán tọa độ trong lưới, chúng ta cần quy đổi các tọa độ này về hệ tọa độ của ảnh gốc. Đảm bảo rằng tọa độ của bounding box được tính dựa trên các cell trong lưới, và nhân với kích thước cell để ra tọa độ trên ảnh gốc.

1. Tính tọa độ tâm bounding box trên ảnh gốc, 32 là số giả dụ cho cellsize.

$$x_{\text{center}} = b_x \cdot 32, \quad y_{\text{center}} = b_y \cdot 32$$

2. Tính tọa độ góc (x_{\min} , y_{\min} , x_{\max} , y_{\max}):

$$x_{\min} = x_{\text{center}} - \frac{b_w}{2}, \quad y_{\min} = y_{\text{center}} - \frac{b_h}{2}$$

$$x_{\max} = x_{\text{center}} + \frac{b_w}{2}, \quad y_{\max} = y_{\text{center}} + \frac{b_h}{2}$$

Bước 5: Tính IoU (Intersection over Union)

Để đánh giá chất lượng của bounding box dự đoán, ta tính toán IoU (Intersection over Union) giữa bounding box dự đoán và bounding box tham chiếu (ground truth).

1. Diện tích giao (A_{inter}):

$$x_{\text{inter}_{\min}} = \max(x_{\min}, x_{\text{ref}_{\min}}), \quad y_{\text{inter}_{\min}} = \max(y_{\min}, y_{\text{ref}_{\min}})$$

$$x_{\text{inter}_{\max}} = \min(x_{\max}, x_{\text{ref}_{\max}}), \quad y_{\text{inter}_{\max}} = \min(y_{\max}, y_{\text{ref}_{\max}})$$

$$A_{\text{inter}} = (x_{\text{inter}_{\max}} - x_{\text{inter}_{\min}}) \cdot (y_{\text{inter}_{\max}} - y_{\text{inter}_{\min}})$$

2. Diện tích hợp (A_{union}):

$$A_{\text{union}} = A_{\text{box}_1} + A_{\text{box}_2} - A_{\text{inter}}$$

3. Tính IoU:

$$\text{IoU} = \frac{A_{\text{inter}}}{A_{\text{union}}}$$

Bước 6: Áp dụng Non-Maximum Suppression (NMS)

Khi có nhiều bounding box dự đoán trùng nhau, chúng ta cần áp dụng NMS (Non-Maximum Suppression) để loại bỏ các bounding box thừa, chỉ giữ lại các bounding box chính xác nhất. Ngưỡng IoU: Nếu IoU giữa một bounding box và bounding box khác lớn hơn ngưỡng > 0.5 , ta sẽ loại bỏ bounding box đó

Kết quả sau khi thực hiện tất cả các bước trên, ta có thể nhận diện được đối tượng trong hình ảnh và xác định chính xác vị trí của nó. Bounding box được dự đoán có tọa độ (xmin, ymin, xmax, ymax) và confidence score p_{obj} , kèm theo nhãn lớp.

2.4.5.2. Áp dụng giải tay thuật toán YOLOv5

Thông qua giới thiệu các công thức sử dụng trong YOLO và YOLOv5, phần này ta sẽ tự tạo ra một bài toán với hình ảnh thu thập được, số liệu tự cho sẵn để giải tay thuật toán YOLOv5, đồng thời sẽ bổ sung thêm các công thức cần thiết phục vụ cho giải bài toán. Ta có 2 bài toán giả định (dễ và khó) như sau:

1. Dễ: Detection Objective về nhận diện chó và mèo trong ảnh bằng mô hình YOLOv5.
2. Khó: Tính bounding box và dự đoán nhãn động vật trong ảnh

Đề bài 1:

Ta có một bức ảnh có kích thước 640x640 chứa một con động vật, trước đó hệ thống đã được train dữ liệu hai động là chó và mèo. Bức ảnh này đã được chia thành 13x13 ô lưới (grid cells).

Mỗi ô lưới sẽ cung cấp các thông tin như sau:

- Bounding Box (x, y, w, h): Vị trí và kích thước của hộp chứa đối tượng.
- Objectness Score (P_{obj}): Xác suất có đối tượng trong ô.
- Class Probabilities (P_{dog} , P_{cat}): Xác suất đối tượng là chó hoặc mèo.

Dưới đây là các kết quả dự đoán của mô hình YOLOv5:

1. Ô lưới (5, 6):
 - Bounding box: ($x = 0.45$, $y = 0.48$, $w = 0.3$, $h = 0.4$)
 - $P_{obj} = 0.9$, $P_{dog} = 0.7$, $P_{cat} = 0.3$
2. Ô lưới (7, 8):
 - Bounding box: ($x = 0.75$, $y = 0.78$, $w = 0.35$, $h = 0.3$)
 - $P_{obj} = 0.85$, $P_{dog} = 0.4$, $P_{cat} = 0.6$
3. Ô lưới (6, 6):
 - Bounding box: ($x = 0.5$, $y = 0.5$, $w = 0.32$, $h = 0.42$)
 - $P_{obj} = 0.6$, $P_{dog} = 0.5$, $P_{cat} = 0.5$

Yêu cầu:

1. Lọc các bounding boxes có $P_{obj} > 0.5$.

2. Xác định đối tượng trong ảnh là Chó hay Mèo dựa trên xác suất P_{dog} và P_{cat} .
3. Áp dụng Non-Maximum Suppression (NMS) để loại bỏ các hộp trùng lặp với $\text{IoU} > 0.5$.

Bài giải

Bước 1: Lọc các bounding boxes có $P_{\text{obj}} > 0.5$

Dựa trên yêu cầu, chúng ta sẽ lọc các ô có P_{obj} nhỏ hơn hoặc bằng 0.5. Cụ thể:

- Ô lưới (5, 6): $P_{\text{obj}} = 0.9 \rightarrow$ Giữ lại.
- Ô lưới (7, 8): $P_{\text{obj}} = 0.85 \rightarrow$ Giữ lại.
- Ô lưới (6, 6): $P_{\text{obj}} = 0.6 \rightarrow$ Giữ lại.

Tất cả các ô đều có $P_{\text{obj}} > 0.5$, do đó không có ô nào bị loại bỏ.

Bước 2: Xác định đối tượng là Chó hay Mèo

- Công thức:

$$P(\text{class}_k) = P_{\text{obj}} \cdot P_{\text{class}_k}$$

Dựa vào xác suất P_{dog} và P_{cat} , ta sẽ phân loại đối tượng trong mỗi ô:

1. Ô lưới (5, 6):
 - $P_{\text{dog}} = 0.7, P_{\text{cat}} = 0.3 \rightarrow$ Dự đoán là Chó vì $P_{\text{dog}} > P_{\text{cat}}$.
2. Ô lưới (7, 8):
 - $P_{\text{dog}} = 0.4, P_{\text{cat}} = 0.6 \rightarrow$ Dự đoán là Mèo vì $P_{\text{cat}} > P_{\text{dog}}$.
3. Ô lưới (6, 6):
 - $P_{\text{dog}} = 0.5, P_{\text{cat}} = 0.5 \rightarrow$ Cả hai xác suất đều bằng nhau, ta có thể chọn Chó hoặc Mèo ngẫu nhiên.

Bước 3: Áp dụng Non-Maximum Suppression

Tính diện tích của các bounding box:

1. Bounding box (5, 6):
 - Vị trí ($x = 0.45, y = 0.48$), kích thước ($w = 0.3, h = 0.4$)
 - Diện tích: $A1 = w \times h = 0.3 \times 0.4 = 0.12$
2. Bounding box (7, 8):
 - Vị trí ($x = 0.75, y = 0.78$), kích thước ($w = 0.35, h = 0.3$)
 - Diện tích: $A2 = w \times h = 0.35 \times 0.3 = 0.105$
3. Bounding box (6, 6):
 - Vị trí ($x = 0.5, y = 0.5$), kích thước ($w = 0.32, h = 0.42$)

- Diện tích: $A3 = w \times h = 0.32 \times 0.42 = 0.1344$

Tính IoU giữa các bounding boxes:

1. IoU giữa (5, 6) và (7, 8):
 - IoU giữa hai hộp này là nhỏ hơn 0.5, do đó chúng không trùng lặp và không cần loại bỏ.
2. IoU giữa (5, 6) và (6, 6):
 - IoU giữa hai hộp này là nhỏ hơn 0.5, do đó chúng không trùng lặp và không cần loại bỏ.
3. IoU giữa (7, 8) và (6, 6):
 - IoU giữa hai hộp này là nhỏ hơn 0.5, do đó chúng không trùng lặp và không cần loại bỏ.

Bước 4: Kết quả sau khi áp dụng NMS

Vì tất cả các hộp không có $\text{IoU} > 0.5$, tất cả các hộp dự đoán đều được giữ lại:

- Bounding box (5, 6): Dự đoán là Chó.
- Bounding box (7, 8): Dự đoán là Mèo.
- Bounding box (6, 6): Dự đoán là Chó hoặc Mèo. Trường hợp số liệu bằng nhau ta sẽ quyết định ngẫu nhiên chọn 1 trong 2 nhãn.

Kết luận: Với các kết quả trên, vẫn chưa nhận diện được nhãn động vật trong hình giả định, do đó ta rút ra được kết luận mô hình chưa được train tốt, các thông số được cung cấp chưa chính xác, tập train chưa đa dạng và cần cải thiện dữ liệu hình ảnh.

Đề bài 2:

Trong bài toán này, ta có một hình ảnh với kích thước 640x540 pixel chứa hình ảnh của một vật thể nằm giữa khung hình. Mô hình YOLOv5 đã được huấn luyện trước đó với 3 anchor boxes:

- Anchor 1: width = 150, height = 150, Confidence score = 0.85, Label = “Mèo”.
- Anchor 2: width = 200, height = 200, Confidence score = 0.6, Label = “Chó”
- Anchor 3: width = 300, height = 300, Confidence score = 0.6, Label = “Mèo”

Các thông số cho trước của hệ thống sau huấn luyện:

1. $\text{gridx}, y = 6$
2. Ngưỡng confidence $P_{obj} > 0.5$
3. Ngưỡng IoU để áp dụng NMS: $\text{IoU} > 0.5$
4. Output từ mô hình:
 - Center offset: $T_x = 0.2, T_y = 0.3$

- Width/Height offset: $Tw = 0.2$, $Th = 0.3$
- Confidence score: $Pobj = 0.7$

Mục tiêu: Tính toán bounding boxes và áp dụng Non-Maximum Suppression (NMS) để chọn ra bounding box cuối cùng và dự đoán nhãn cho vật thể.

Bài giải

Bước 1: Resize và chuẩn hóa ảnh

Được biết ta có hình ảnh gốc có kích thước 500x421, ta cần resize hình ảnh về kích thước chuẩn. Áp dụng công thức ta tính tỷ lệ giữa chiều dài và chiều rộng:

$$- S = \min(640/500, 640/421) \Rightarrow S = \min(1.28, 1.52) = 1.28$$

Ta có $S_{min} = 1.28$, do đó chiều cao và chiều rộng sau khi resize sẽ là:

- $H_{new} = 421 \times 1.28 \approx 540$ (px)
- $W_{new} = 500 \times 1.28 = 640$ (px)

Vậy, sau khi resize ta có hình ảnh với kích thước là 640x540 (px).

Bước 2: Chia ảnh thành lưới (Grid)

Cấu hình YOLO có giá trị phổ biến là 13x13, 19x19 và 23x23. Ở đây, ta sẽ chia ảnh 640x540 thành 13x13 cells:

- Chiều ngang(width): $640/13 \approx 49.23$ pixels
- Chiều cao (height): $540/13 \approx 41.54$ pixels

Như vậy, kích thước cell cho grid 13x13 sẽ là khoảng 49x42 pixels.



Figure 16. Ảnh sau khi tính resize và cells

Bước 3: Tính toán các bounding boxes từ các anchor boxes

Áp dụng công thức sau để tính tọa độ trung tâm Cx và Cy:

$$c_x = (Tx + grid_x) \times cell_size_x$$

$$c_y = (Ty + grid_y) \times cell_size_y$$

1. Tính tọa độ Cx và Cy:

- $Cx = (Tx + grid_x) \times cell_size_x = (0.2 + 6) \times 49.23 = 6.2 \times 49.23 = 305.43px$
- $Cy = (Ty + grid_y) \times cell_size_y = (0.3 + 6) \times 41.54 = 6.3 \times 41.54 = 261.70px$

2. Tính toán kích thước của bounding box:

Với Anchor 1 (width = 150, height = 150):

- $W1 = 150 \times e(Tw) = 150 \times e(0.2) \approx 150 \times 1.221 = 183.15px$
- $H1 = 150 \times e(Th) = 150 \times e(0.3) \approx 150 \times 1.349 = 202.35px$

Với Anchor 2 (width = 200, height = 200):

- $W2 = 200 \times e(Tw) = 200 \times e(0.2) \approx 200 \times 1.221 = 244.20px$
- $H2 = 200 \times e(Th) = 200 \times e(0.3) \approx 200 \times 1.349 = 269.80px$

Với Anchor 3 (width = 300, height = 300):

- $W3 = 300 \times e(Tw) = 300 \times e(0.2) \approx 300 \times 1.221 = 366.30px$
- $H3 = 300 \times e(Th) = 300 \times e(0.3) \approx 300 \times 1.349 = 404.70px$

Bước 4: Tính toán bounding box cho từng anchor box

1. Bounding box từ Anchor 1:

- Left: $305.43 - 183.15/2 = 305.43 - 91.57 = 213.86 px$
- Top: $261.70 - 202.35/2 = 261.70 - 101.18 = 160.52 px$
- Right: $305.43 + 183.15/2 = 305.43 + 91.57 = 397.00 px$
- Bottom: $261.70 + 202.35/2 = 261.70 + 101.18 = 362.88 px$

2. Bounding box từ Anchor 2:

- Left: $305.43 - 244.20/2 = 305.43 - 122.10 = 183.33 px$
- Top: $261.70 - 269.80/2 = 261.70 - 134.90 = 126.80 px$
- Right: $305.43 + 244.20/2 = 305.43 + 122.10 = 427.53 px$
- Bottom: $261.70 + 269.80/2 = 261.70 + 134.90 = 396.60 px$

3. Bounding box từ Anchor 3:

- Left: $305.43 - 366.30/2 = 305.43 - 183.15 = 122.28 px$
- Top: $261.70 - 404.70/2 = 261.70 - 202.35 = 59.3 px$
- Right: $305.43 + 366.30/2 = 305.43 + 183.15 = 488.58 px$
- Bottom: $261.70 + 404.70/2 = 261.70 + 202.35 = 464.05 px$

Bước 5: Tính IoU giữa các bounding boxes từ các anchor boxes

5.1. Tính toán IoU giữa Anchor 1 và Anchor 2

1. Diện tích giao: Tính toán diện tích giao giữa box từ Anchor 1 và Anchor 2.

- Intersection width: $\max(0, \min(397.00, 427.53) - \max(213.86, 183.33)) = \max(0, 397.00 - 213.86) = 183.14 \text{ px}$
- Intersection height: $\max(0, \min(362.88, 396.60) - \max(160.52, 126.80)) = \max(0, 362.88 - 160.52) = 202.36 \text{ px}$

Ta có diện tích giao: Intersection Area = $183.14 \times 202.36 = 37090.73 \text{ px}^2$

2. Diện tích hợp:

- Area of Box A = $(397.00 - 213.86) \times (362.88 - 160.52) = 183.14 \times 202.36 = 37090.73 \text{ px}^2$
- Area of Box B = $(427.53 - 183.33) \times (396.60 - 126.80) = 244.20 \times 269.80 = 65983.16 \text{ px}^2$
- Union Area = $37090.73 + 65983.16 - 37090.73 = 65983.16 \text{ px}^2$

3. IoU:

- $\text{IoU} = 37090.73 / 65983.16 = 0.56$

5.2. Tính toán IoU giữa Anchor 1 và Anchor 3

1. Diện tích giao: Tính toán diện tích giao giữa box từ Anchor 1 và Anchor 3.

- Intersection width: $\max(0, \min(397.00, 488.58) - \max(213.86, 122.28)) = \max(0, 397.00 - 213.86) = 183.14 \text{ px}$
- Intersection height: $\max(0, \min(362.88, 464.05) - \max(160.52, 59.35)) = \max(0, 362.88 - 160.52) = 202.36 \text{ px}$

Ta có diện tích giao: Intersection Area = $183.14 \times 202.36 = 37090.73 \text{ px}^2$

2. Diện tích hợp:

- Area of Box A = $(397.00 - 213.86) \times (362.88 - 160.52) = 183.14 \times 202.36 = 37090.73 \text{ px}^2$
- Area of Box B = $(488.58 - 122.28) \times (464.05 - 59.35) = 366.30 \times 404.70 = 148091.91 \text{ px}^2$
- Union Area = $37090.73 + 148091.91 - 37090.73 = 148091.91 \text{ px}^2$

3. IoU:

- $\text{IoU} = 37090.73 / 148091.91 = 0.25$

5.3. Tính toán IoU giữa Anchor 1 và Anchor 3

1. Diện tích giao: Tính toán diện tích giao giữa box từ Anchor 1 và Anchor 3.

- Intersection width: $\max(0, \min(427.53, 488.58) - \max(183.33, 122.28)) = \max(0, 427.53 - 183.33) = 244.20 \text{ px}$
- Intersection height: $\max(0, \min(396.60, 464.05) - \max(126.80, 59.35)) = \max(0, 396.60 - 126.80) = 269.80 \text{ px}$

Ta có diện tích giao: $\text{Intersection Area} = 244.20 \times 269.80 = 65983.16 \text{ px}^2$

2. Diện tích hợp:

- Area of Box A = $(397.00 - 213.86) \times (362.88 - 160.52) = 183.14 \times 202.36 = 37090.73 \text{ px}^2$
- Area of Box B = $(488.58 - 122.28) \times (464.05 - 59.35) = 366.30 \times 404.70 = 148091.91 \text{ px}^2$
- Union Area = $37090.73 + 148091.91 - 65983.16 = 148099.58 \text{ px}^2$

3. IoU:

- $\text{IoU} = 65983.16 / 148099.58 = 0.445$

Bước 6: Áp dụng Non-Maximum Suppression (NMS)

Với các IoU tính được:

- IoU giữa Anchor 1 và Anchor 2 là 0.56, lớn hơn 0.5, nên bạn nên loại bỏ Anchor 2 vì sự chồng lấn quá lớn.
- IoU giữa Anchor 1 và Anchor 3 là 0.25, nhỏ hơn 0.5, không cần loại bỏ Anchor 3.

Tuy nhiên, confidence score của các anchor boxes là:

- Anchor 1: 0.85 (cao nhất)
- Anchor 2: 0.60
- Anchor 3: 0.65

Vì vậy, Anchor 1 với confidence score cao nhất (0.85) được giữ lại, mặc dù có một sự chồng lấn nhỏ với Anchor 2, và Anchor 3 bị loại bỏ vì có confidence score thấp hơn.

Kết luận: Bounding box cuối cùng giữ lại là Anchor 1 với nhãn là “Mèo”, vì nó có confidence score cao nhất và không bị loại do IoU nhỏ hơn 0.5 với Anchor 3. Bounding box cuối cùng Anchor 1 tọa độ:

- Left: 213.86 px
- Top: 160.52 px
- Right: 397.00 px
- Bottom: 362.88 px

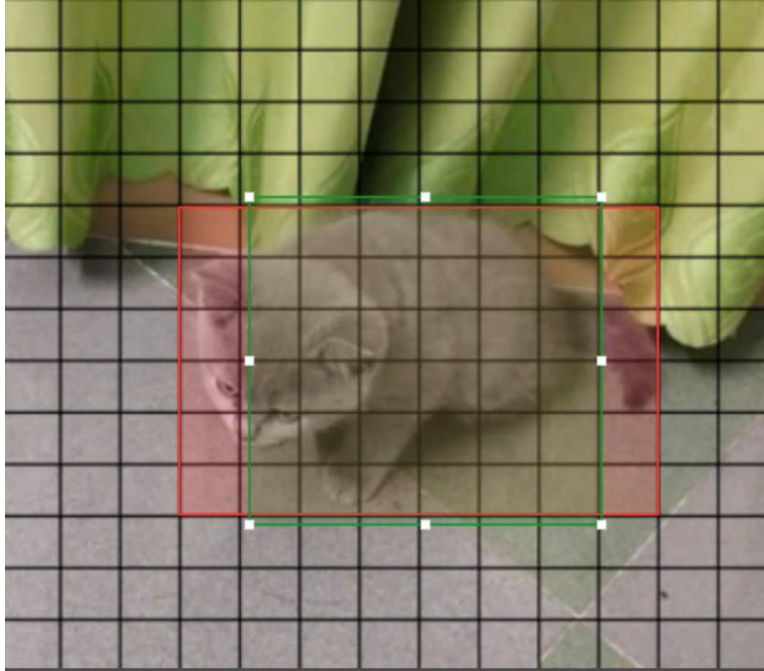


Figure 17. Kết quả minh họa của tìm bounding box

CHƯƠNG 3: CHUẨN BỊ DỮ LIỆU

Quá trình chuẩn bị dữ liệu là một bước quan trọng và quyết định hiệu quả của việc huấn luyện mô hình nhận diện vật thể bằng YOLOv5. Mục tiêu của bước này là xây dựng một bộ dữ liệu chất lượng, đầy đủ và phù hợp với yêu cầu của thuật toán. Quy trình bao gồm các giai đoạn từ thu thập dữ liệu, khám phá, làm sạch, đến gia tăng độ phong phú và lưu trữ dữ liệu để đảm bảo dữ liệu sẵn sàng cho quá trình huấn luyện.'

3.1. Thu thập dữ liệu

Quá trình thu thập dữ liệu được thực hiện từ nhiều nguồn đáng tin cậy, đảm bảo cung cấp đủ thông tin cho việc huấn luyện mô hình YOLOv5. Cụ thể:

1. Dataset coco128: Được lấy từ Github bằng cách tìm kiếm từ khóa "coco128". Bộ dữ liệu này bao gồm các hình ảnh đã được gắn nhãn, hỗ trợ nhận diện gần 80 loại vật thể khác nhau. Link tải được sao chép trực tiếp từ repository.
2. Dữ liệu cá nhân: Tự tạo thông qua tự chụp và gắn nhãn thủ công.

```

73 56: chair
74 57: couch
75 58: potted plant
76 59: bed
77 60: dining table
78 61: toilet
79 62: tv
80 63: laptop
81 64: mouse
82 65: remote
83 66: keyboard
84 67: cell phone
85 68: microwave
86 69: oven
87 70: toaster
88 71: sink
89 72: refrigerator
90 73: book
91 74: clock
92 75: vase
93 76: scissors
94 77: teddy bear
95 78: hair drier
96 79: toothbrush
97
98 # Download script/URL (optional)
99 download: https://github.com/ultraalytics/assets/releases/download/v0.0.0/coco128.zip

```

Figure 18. Link tải dataset và số lượng đối tượng nhận diện trên Github



Figure 19. Ảnh khuôn mặt tự tạo

p (2).txt	1/15/2025 11:01 AM	Text Document	1 KB
p (3).txt	1/15/2025 11:01 AM	Text Document	1 KB
p (4).txt	1/15/2025 11:01 AM	Text Document	1 KB
p (5).txt	1/15/2025 11:01 AM	Text Document	1 KB
p (6).txt	1/15/2025 11:01 AM	Text Document	1 KB
p (7).txt	1/15/2025 11:01 AM	Text Document	1 KB

Figure 20. Nhân được gắn của ảnh khuôn mặt

3.2. Khám phá và đánh giá dữ liệu

- Dataset coco128: Bộ dữ liệu này đã được tổ chức tốt với các tập huấn luyện và nhãn đầy đủ, đáp ứng yêu cầu đầu vào của thuật toán YOLOv5.
- Dataset cá nhân: Đây là dữ liệu tự thu thập, chưa được gắn nhãn ban đầu. Việc gắn nhãn thủ công cần được thực hiện để đảm bảo tính đồng nhất và chính xác trước khi đưa vào sử dụng.

3.3. Làm sạch và xác thực dữ liệu

- Dữ liệu coco128: Do bộ dữ liệu này được lấy từ nguồn uy tín và đã qua xử lý ban đầu, việc làm sạch chủ yếu tập trung vào kiểm tra lại tính đầy đủ và loại bỏ các ngoại lệ nếu có.
- Dữ liệu cá nhân: Là dữ liệu tự tạo, cần gắn nhãn nên sẽ làm thủ công bằng các bước sau:
 1. Tự chụp các góc độ mặt để có cái nhìn đa dạng nhất.
 2. Gắn nhãn thủ công toàn bộ dữ liệu bằng công cụ MakeSense.AI, đảm bảo mỗi hình ảnh đều có nhãn phù hợp với mục tiêu nhận diện.

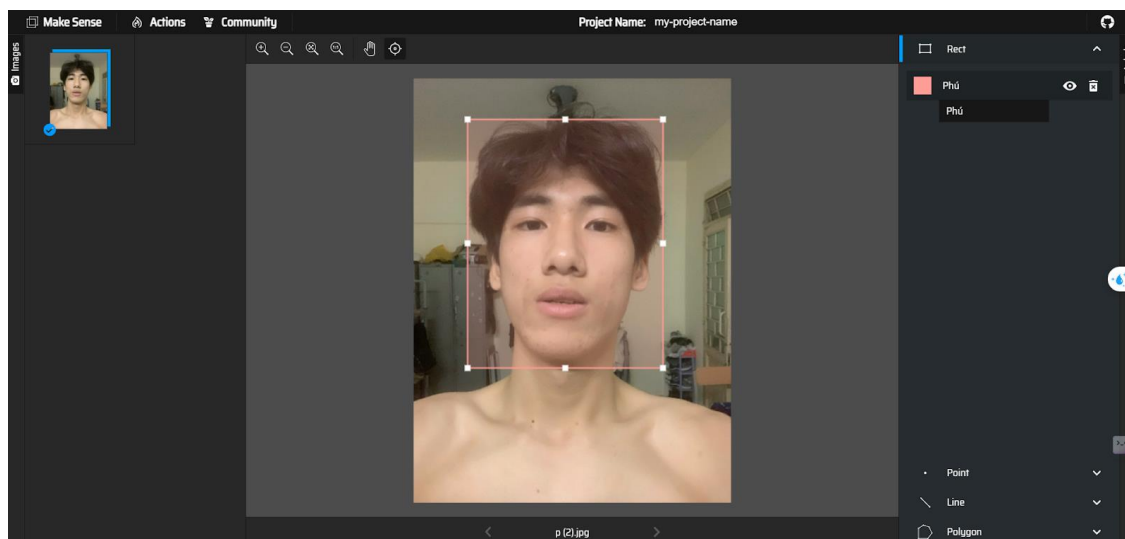


Figure 21. Gắn nhãn cho hình ảnh cá nhân

3.4. Lưu trữ dữ liệu

- Sau khi hoàn tất các bước chuẩn bị, toàn bộ dữ liệu được tổ chức và lưu trữ trong các thư mục riêng biệt theo cấu trúc yêu cầu của YOLOv5 (bao gồm train và labels).
- Dữ liệu đã qua xử lý sẵn sàng được sử dụng cho quá trình huấn luyện mô hình trong các bước tiếp theo.

CHƯƠNG 4: CÀI ĐẶT CHƯƠNG TRÌNH

Ở phần này, cài đặt và thí nghiệm sẽ dùng trên cả hai dữ liệu và khi áp dụng thực tế sẽ chỉ dùng dataset từ github.

4.1. Quá trình huấn luyện đối với dữ liệu lớn thu thập

Vì dữ liệu lớn đã có các dataset và đã được chủ sở hữu train trước đó nên chỉ cần code các mã phù hợp và detect là nhận diện được.

Bước 1: Cài đặt YOLOv5 vào Google Colab với dữ liệu lớn

Sử dụng nền tảng có sẵn trên Google Colab kết hợp với models YOLOv5 để tiến hành quá trình huấn luyện

```
✓ 41s [1] !git clone https://github.com/ultralytics/yolov5 # clone
      %cd yolov5
      %pip install -qr requirements.txt comet_ml # install

      import torch
      import utils
      display = utils.notebook_init() # checks
```

➡ YOLOv5 🚀 v7.0-395-g6420a1db Python-3.10.12 torch-2.5.1+cu121 CPU
Setup complete ✅ (2 CPUs, 12.7 GB RAM, 32.6/107.7 GB disk)

Figure 22. Clone YOLOv5 và cài đặt các dependencies

Bước 2: Thí nghiệm nhận diện dựa trên dữ liệu lớn có sẵn

Tải hình ảnh mình muốn nhận diện và upload lên, sau đó copy path gắn vào phía sau để detect vật thể trong hình.

```
✓ 14s !python detect.py --source /content/traidepphuyen.jpg

➡ detect: weights=yolov5s.pt, source=/content/traidepphuyen.jpg, data=data/coco128.yaml, imgsz=[640, 640],
YOLOv5 🚀 v7.0-395-g6420a1db Python-3.10.12 torch-2.5.1+cu121 CPU

Downloading https://github.com/ultralytics/yolov5/releases/download/v7.0/yolov5s.pt to yolov5s.pt...
100% 14.1M/14.1M [00:00<00:00, 146MB/s]

Fusing layers...
YOLOv5s summary: 213 layers, 7225885 parameters, 0 gradients, 16.4 GFLOPs
image 1/1 /content/traidepphuyen.jpg: 640x544 1 person, 467.8ms
Speed: 5.6ms pre-process, 467.8ms inference, 35.9ms NMS per image at shape (1, 3, 640, 640)
Results saved to runs/detect/exp
```

Figure 23. Copy path và detect hình ảnh

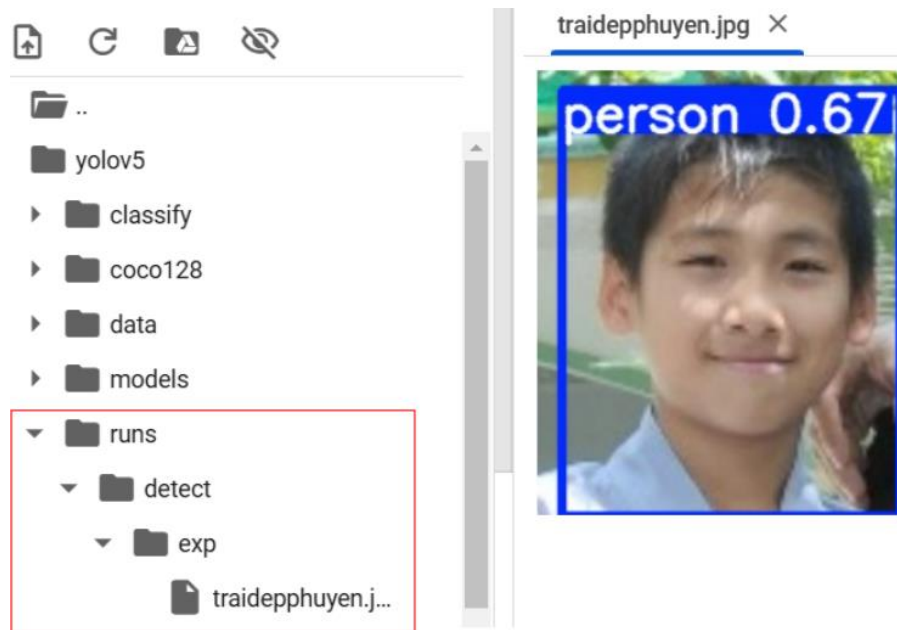


Figure 24. Ảnh nhận diện được detect và lưu trữ trong tệp run

Kết quả: Sau khi lấy các hình ảnh bất kỳ trên mạng và thử nghiệm bằng cách detect và tất cả đều được nhận diện đúng vì đều nằm trong các vật đã được train sẵn trước đó.



Figure 25. Kết quả detect đa hình ảnh

4.2. Quá trình huấn luyện đối với dữ liệu cá nhân

Bước 1: Clone và Install YOLOv5 từ Github

```
✓ 7s [2] !git clone https://github.com/ultralytics/yolov5 # clone
      %cd yolov5
      %pip install -qr /content/yolov5/requirements.txt comet_ml # install

      import torch
      import utils
      display = utils.notebook_init() # checks
```

Figure 26. Clone và tải YOLOv5

Bước 2: Upload file coco128 và unrar

Ở bước này, ta sẽ xóa các file dữ liệu đã tải từ github và thay thế đó là ảnh và nhãn cá nhân sau đó nén rồi upload lên Colab.

```
✓ 5s [2] !unrar x /content/coco128.rar

Show hidden output
```

Figure 27. Unrar coco128

Bước 3: Đổi số lượng và tên gọi trong coco128.yaml

Đổi tên gốc của file coco128.yaml sang tên các đối tượng cá nhân, đó là 0: 'Phú'. Đồng thời copy path vào các mục train và val.

```
coco128.yaml X
1
2 train: /content/yolov5/yolov5/coco128/images/train2017 # train images (relative to 'path') 128 images
3 val: /content/yolov5/yolov5/coco128/images/train2017 # val images (relative to 'path') 128 images
4 test: # test images (optional)
5
6 # Classes
7 nc: 1
8 names:
9   0: 'Phu'
10
```

Figure 28. Gán đường dẫn vào tệp coco128.yaml

Bước 4: Thực hiện lệnh train cho mô hình

```
✓ 49m [12] !python /content/yolov5/train.py --img 640 --epochs 50 --data coco128.yaml --weights yolov5s.pt

Show hidden output
```

Figure 29. Code tập train

Sau khi train mô hình xong, hệ thống sẽ cung cấp cho chúng ta bảng đánh giá kết quả như bên dưới, nhìn sơ qua ta thấy được càng nâng số lần epoch lên thì độ chính xác của mô hình sẽ tăng lên.

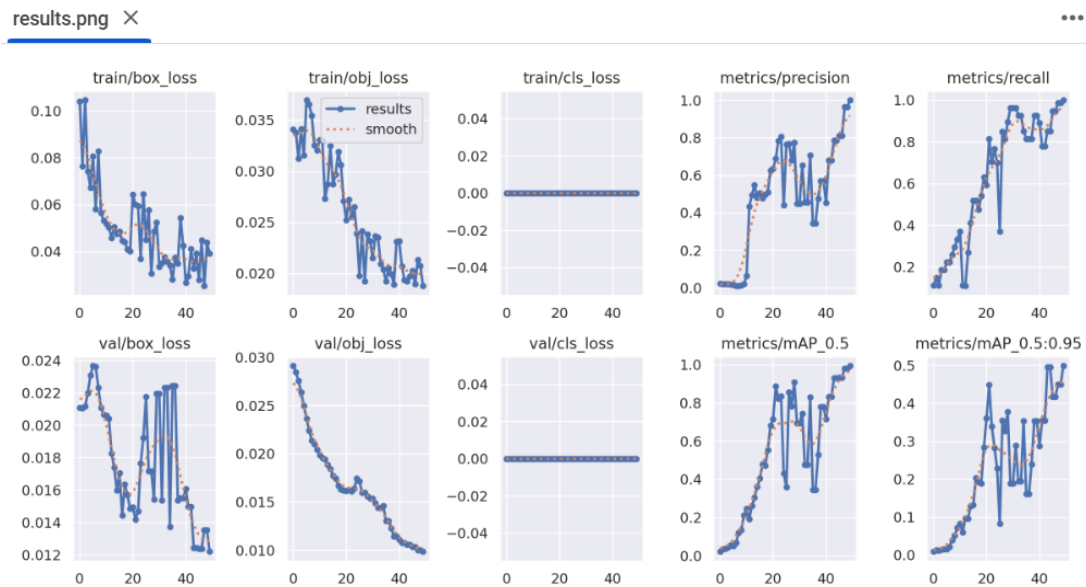


Figure 30. Hiệu số khi được epoch 50 lần

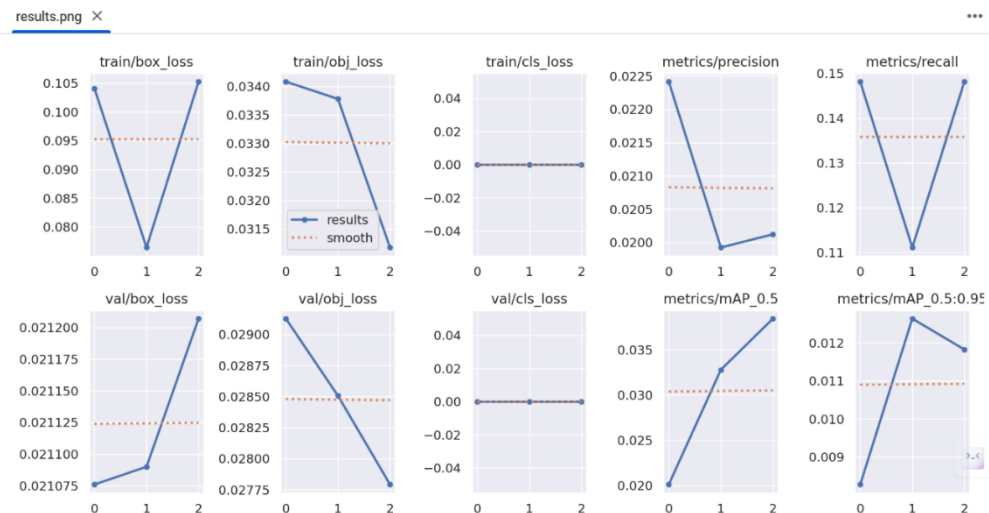


Figure 31. Hiệu số khi được epoch 3 lần

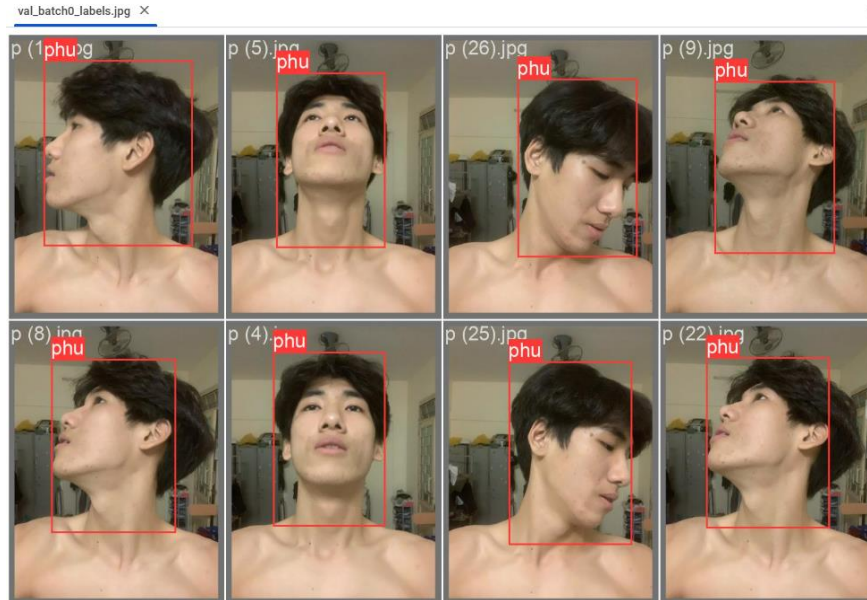


Figure 32. Giá trị được dán nhãn trong tệp val_batch0_labels.jpg

Bước 5: Thực hiện detect

Trước tiên, dán tệp best.pt thay thế cho chữ yolov5s.pt trong code file detect.py.

```
parser = argparse.ArgumentParser()
parser.add_argument("--weights", nargs="+", type=str, default=ROOT / "/content/yolov5/runs/train/exp2/weights/best.pt")
parser.add_argument("--source", type=str, default=ROOT / "data/images", help="file/dir/URL/glob/screen/0(webcam)")
parser.add_argument("--data", type=str, default=ROOT / "data/coco128.yaml", help="(optional) dataset.yaml path")
parser.add_argument("--imgsz", "--img", "--img-size", nargs="+", type=int, default=[640], help="inference size h,w")
parser.add_argument("--conf-thres", type=float, default=0.25, help="confidence threshold")
parser.add_argument("--iou-thres", type=float, default=0.45, help="NMS IoU threshold")
parser.add_argument("--max-det", type=int, default=1000, help="maximum detections per image")
parser.add_argument("--device", default="", help="cuda device, i.e. 0 or 0,1,2,3 or cpu")
```

Figure 33. Copy path vào file detect

Sau đó, copy path tệp detect.py và tệp ảnh vào code để thực hiện nhận diện.

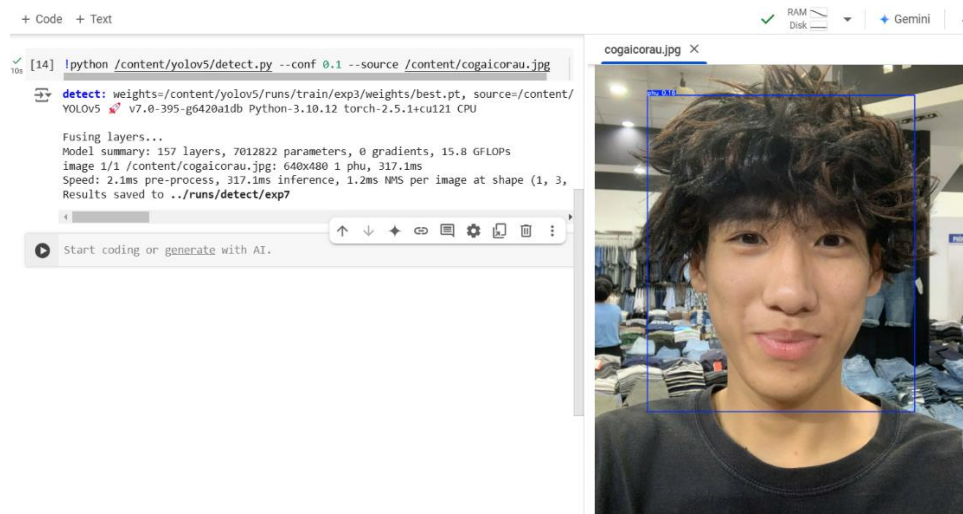


Figure 34. Code và kết quả detect

Bước 6: Đánh giá kết quả

Với số lượng ảnh là gần 30 và được chụp ở nhiều góc độ, cùng số lần train 50. Vì thế, hạn chế ở số lần và số lượng train nên hình ảnh dự đoán không lớn hơn 0.5 confidence và có hình xuất hiện sai số với nhiều bounding box. Tuy nhiên, ở bước này chỉ là thử nghiệm và kết quả cũng cho thấy thuật toán này hiệu quả trong quá trình nhận diện đối tượng, có thể cải thiện biến mô hình hiệu quả hơn bằng số lượng và số lần cho tập train.

Ngoài ra, có thể gia tăng confidence bằng cách nâng số lần epoch lên, minh chứng như 2 hình hiệu số ở trên cho ta thấy, confidence tăng tỉ lệ thuận với số lần epoch cho tập train.

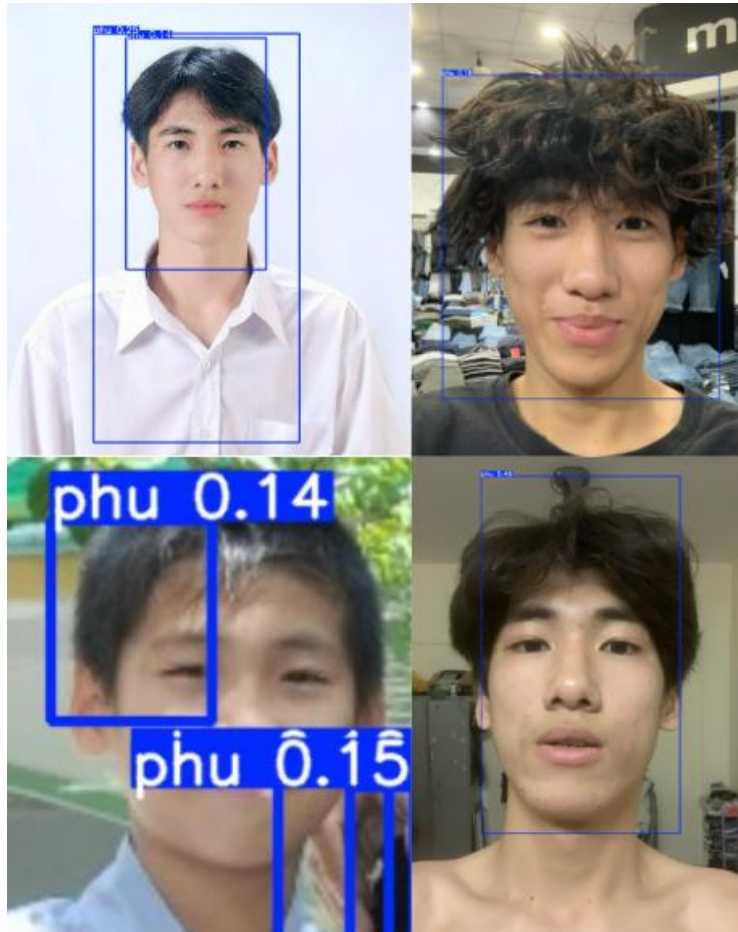


Figure 35. Số confidence mô hình dự đoán cho các tệp ảnh

CHƯƠNG 5: THIẾT KẾ HỆ THỐNG

Đối với thiết kế hệ thống, ta sẽ dùng phần mềm Visual Studio Code để chạy chương trình cũng như thiết kế web. Những bước thực hiện lần lượt là tạo folder, sắp xếp các tệp, code trên file đuôi html và py cuối cùng chạy pip trên Terminal và thực hành.

5.1. Thiết kế website nhận diện

Bước 1: Tạo và sắp xếp file

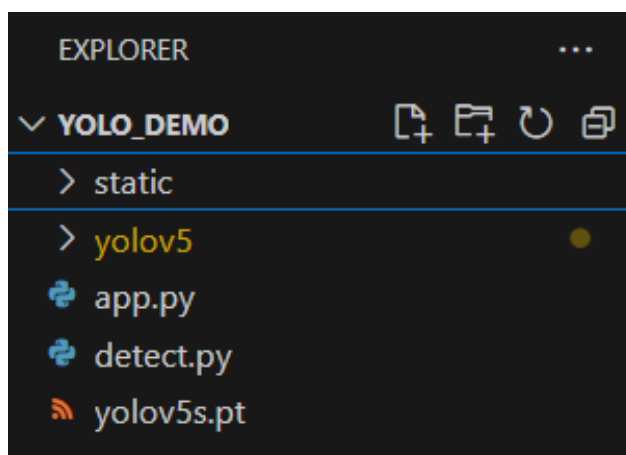


Figure 36. Tạo file

Bước 2: Cài đặt YOLOv5 và chạy các thư viện

```
PS C:\yolo_demo> pip install flask torch torchvision opencv-python
>> git clone https://github.com/ultralytics/yolov5.git
>> cd yolov5
>> pip install -r requirements.txt
```

Figure 37. Install thư viện và YOLOv5

Bước 3: Code giao diện và cho app.py

```
app.py 5 x index.html U
yolov5 > app.py > ...
1 import os
2 import torch
3 from flask import Flask, render_template, request, redirect, url_for
4 from werkzeug.utils import secure_filename
5 import cv2
6 import numpy as np
7
8 app = Flask(__name__)
9
10 # Đảm bảo thư mục 'uploads' tồn tại trong 'static'
11 UPLOAD_FOLDER = 'static/uploads'
12 if not os.path.exists(UPLOAD_FOLDER):
13     os.makedirs(UPLOAD_FOLDER)
14 app.config['UPLOAD_FOLDER'] = UPLOAD_FOLDER
15 ALLOWED_EXTENSIONS = ['png', 'jpg', 'jpeg']
16
17 # Kiểm tra định dạng file ảnh
18 def allowed_file(filename):
19     return '.' in filename and filename.rsplit('.', 1)[1].lower() in ALLOWED_EXTENSIONS
20
21 # Tải mô hình YOLOv5
22 model = torch.hub.load('ultralytics/yolov5', 'yolov5s')
```

Figure 38. Code của app.py

```
app.py 5 index.html U x
yolov5 > templates > index.html > html > head > style > .card
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4     <meta charset="UTF-8">
5     <meta name="viewport" content="width=device-width, initial-scale=1.0">
6     <title>Object Detection with YOLOv5</title>
7     <!-- Thêm Bootstrap 5 và FontAwesome -->
8     <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/css/bootstrap.min.css" rel="stylesheet">
9     <link href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/6.0.0-beta3/css/all.min.css" rel="stylesheet">
10 </head>
11 <body>
12     <div class="upload-container">
13         <div class="img-result">
14             <img alt="Placeholder for image result" />
15         </div>
16     </div>
17 </body>
18 </html>
```

Figure 39. Thiết kế giao diện

Bước 4: Chạy python app.py

```
PS C:\yolo_demo> python app.py
Using cache found in C:\Users\ADMIN\.cache\torch\hub\ultralytics_yolov5_master
YOLOv5 2025-1-14 Python-3.9.5 torch-2.5.1+cpu CPU
Downloading https://github.com/ultralytics/yolov5/releases/download/v7.0/yolov5s.pt to yolov5s.pt...
100%|
```

Figure 40. Chạy chương trình

Bước 5: Theo đường link và sử dụng website

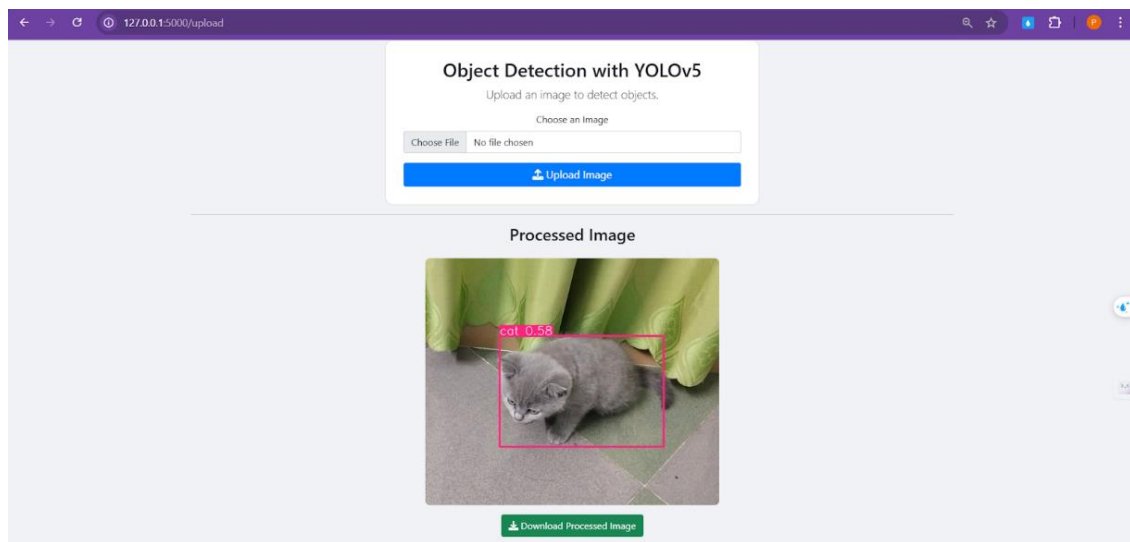


Figure 41. Kết quả nhận diện trên website

Khi người dùng truy cập và upload file, ảnh người dùng sẽ được lưu trữ trong file của người sản xuất, ngoài ra người dùng có thể download hình ảnh nếu muốn.

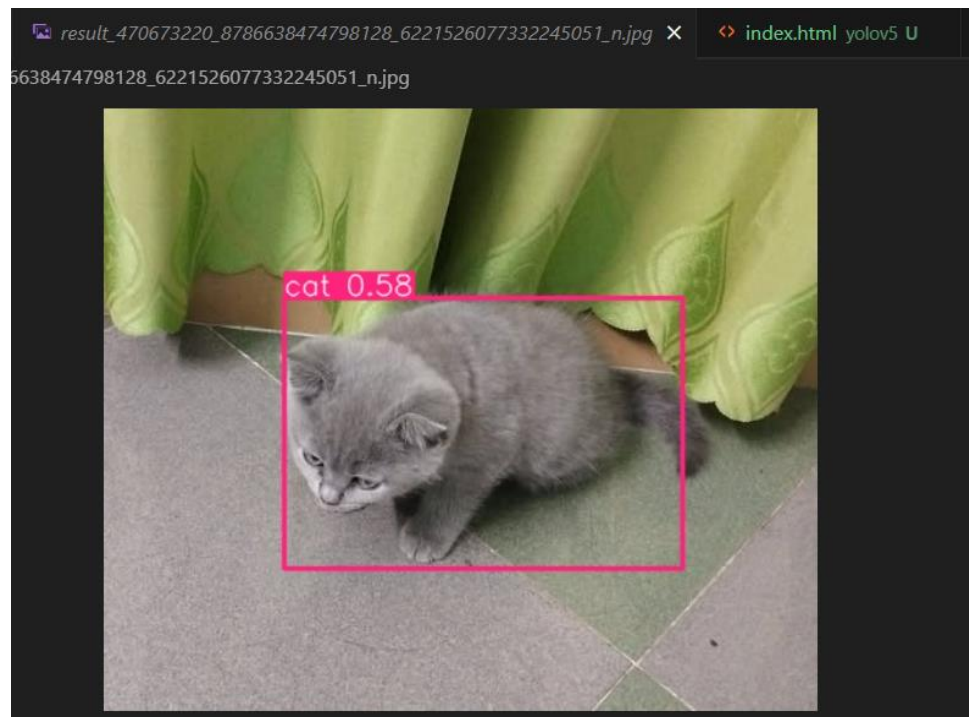
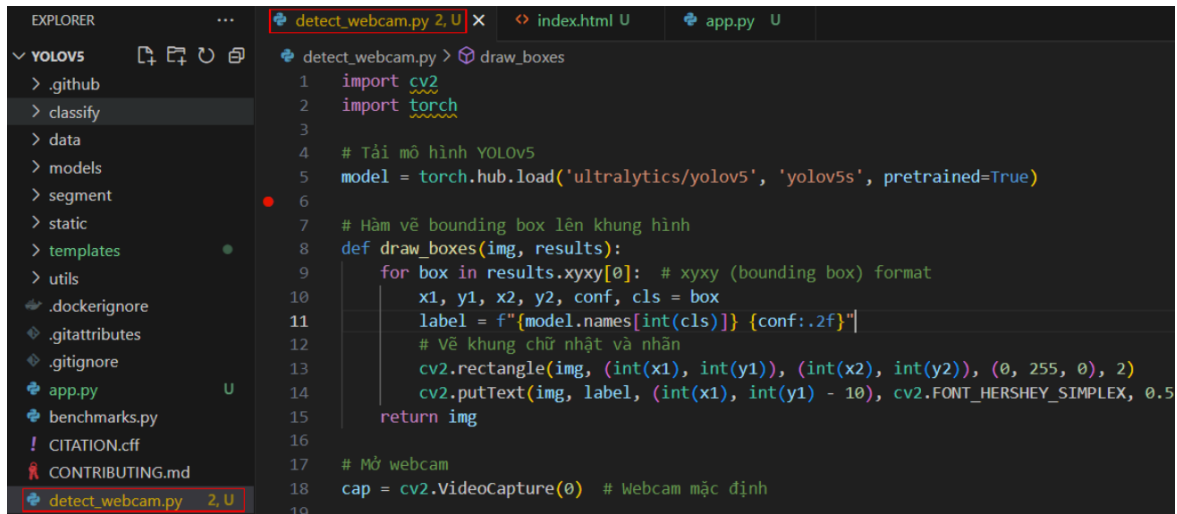


Figure 42. Ảnh người dùng upload để nhận diện được gửi về hệ thống

5.2. Tạo demo webcam nhận diện

Bước 1: Tạo tệp detect_webcam.py trong tệp YOLOv5 và code tạo webcam



```
1 import cv2
2 import torch
3
4 # Tải mô hình YOLOv5
5 model = torch.hub.load('ultralytics/yolov5', 'yolov5s', pretrained=True)
6
7 # Hàm vẽ bounding box lên khung hình
8 def draw_boxes(img, results):
9     for box in results.xyxy[0]: # xyxy (bounding box) format
10         x1, y1, x2, y2, conf, cls = box
11         label = f"{model.names[int(cls)]} {conf:.2f}"
12         # Vẽ khung chữ nhật và nhãn
13         cv2.rectangle(img, (int(x1), int(y1)), (int(x2), int(y2)), (0, 255, 0), 2)
14         cv2.putText(img, label, (int(x1), int(y1) - 10), cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 255, 0))
15     return img
16
17 # Mở webcam
18 cap = cv2.VideoCapture(0) # Webcam mặc định
19
```

Figure 43. Code của tệp detect_webcam.py

Bước 2: Chạy Terminal để hiện webcam với code python detect_webcam.py

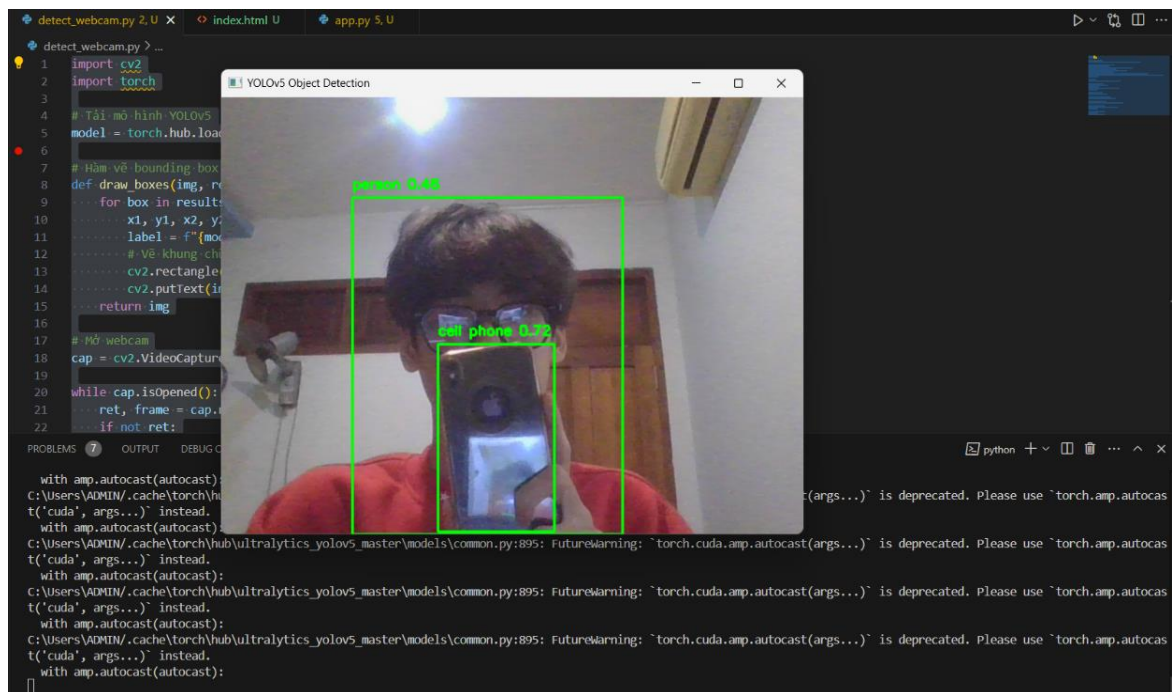


Figure 44. Sử dụng webcam để nhận diện

Kết luận: Sử dụng webcam dựa trên bộ dataset coco128 đã nhận diện tốt các đối tượng xuất hiện qua webcam, đối với đối tượng điện thoại có confidence khá cao >0.5 và dù person thấp hơn 0.5 vì bị che mặt nhưng mô hình vẫn nhận diện tốt.

CHƯƠNG 6: BÁO CÁO KẾT QUẢ

6.1. Tổng kết kết quả thực nghiệm

6.1.1. Mô tả quá trình thực nghiệm

Mục tiêu chính của thực nghiệm là kiểm tra hiệu quả của thuật toán YOLOv5 trên hai bộ dữ liệu:

- Bộ dữ liệu chuẩn (coco128): Dữ liệu có sẵn từ YOLOv5, bao gồm 128 hình ảnh được gắn nhãn đầy đủ với nhiều loại đối tượng khác nhau.
- Bộ dữ liệu tự tạo: Bao gồm 30 hình ảnh được thu thập và gắn nhãn thủ công, chủ yếu tập trung vào một số loại đối tượng cụ thể.

Các bước thực nghiệm:

- Huấn luyện mô hình trên cả bộ dữ liệu cá nhân với số lần epoch là 50.
- So sánh độ chính xác, tốc độ nhận diện và mức độ phù hợp của thuật toán với từng loại dữ liệu.
- Đánh giá kết quả dựa trên các chỉ số: mAP (mean Average Precision), Precision, Recall, và Confidence score.

6.1.2. Bảng tổng hợp kết quả thí nghiệm khi train 2 dataset

Chỉ số	Coco128	Dataset cá nhân
Precision (P)	0.736	1.0
Recall ®	0.637	1.0
mAP50	0.727	0.995
Map50-95	0.484	0.499
Số Epoch	3	50
Thời gian huấn luyện	0.248 giờ	0.792 giờ

Table 3. Bảng tổng hợp kết quả thí nghiệm khi train 2 dataset

Nhận xét:

- Dataset cá nhân có hiệu suất vượt trội với Precision và Recall đạt 1.0, nhưng chỉ số mAP50-95 vẫn chưa đủ cao. Tuy nhiên, việc dataset cá nhân chỉ gồm 27 hình ảnh với số lượng đối tượng là 1 có thể là lý do các chỉ số “đẹp” hơn dataset của Github.
- Coco128 có sự cân bằng tốt hơn giữa Precision và Recall, nhưng không đạt hiệu quả cao trong các chỉ số mAP50-95, điều này cho thấy mô hình có thể chưa tối ưu khi đối mặt với các tình huống phức tạp hơn.

6.1.3. Đánh giá hiệu quả thuật toán

Ưu điểm:

- YOLOv5 nhận diện nhanh, hiệu quả tốt trên dữ liệu lớn.
- Thuật toán duy trì hiệu suất cao ngay cả trên các thiết bị phần cứng trung bình như Google Colab.

Hạn chế:

- Hiệu quả suy giảm rõ rệt khi dữ liệu nhỏ hoặc không đồng nhất.
- Các đối tượng bị che khuất hoặc có kích thước quá nhỏ thường không được nhận diện chính xác.

6.2. Nhận xét và đánh giá

6.2.1. Điểm mạnh của thuật toán YOLOv5

- Tốc độ nhanh: YOLOv5 thực hiện nhận diện theo thời gian thực, vượt trội so với nhiều thuật toán khác như Faster R-CNN hay SSD.
- Độ chính xác cao: Với các chỉ số Precision và Recall vượt trên 50% trong thực nghiệm với coco128.
- Khả năng tùy chỉnh: Hỗ trợ nhiều loại cấu trúc mạng như YOLOv5s, YOLOv5m, YOLOv5l, YOLOv5x, phù hợp với các bài toán khác nhau.

6.2.2. Điểm yếu của thuật toán YOLOv5

- Nhạy cảm với chất lượng dữ liệu: Yêu cầu dữ liệu được gán nhãn cẩn thận và đa dạng.
- Tài nguyên huấn luyện: Đòi hỏi bộ xử lý mạnh (RAM, GPU) nếu huấn luyện trên bộ dữ liệu lớn.
- Hạn chế với đối tượng nhỏ: Kích thước nhỏ hoặc bị che khuất ảnh hưởng đáng kể đến hiệu suất.

6.2.3. Đề xuất cải tiến và mở rộng

1. Tăng cường dữ liệu:
 - Thu thập thêm hình ảnh và thực hiện data augmentation để gia tăng đa dạng.
 - Kết hợp các bộ dữ liệu khác như Pascal VOC, Open Images.
2. Điều chỉnh mô hình:
 - Sử dụng YOLOv5x (phiên bản lớn nhất của YOLOv5) để cải thiện hiệu suất hoặc sử dụng các phiên bản YOLOv mới nhất.
 - Áp dụng transfer learning từ các mô hình pretrained lớn.
3. Tích hợp công nghệ khác:
 - Kết hợp YOLOv5 với các thuật toán xử lý ảnh khác để xử lý các đối tượng bị che khuất.
 - Tăng cường mô hình bằng cách kết hợp YOLO với mạng Transformer.

6.3. Kết luận và hướng phát triển

6.3.1. Kết luận

Đồ án đã đạt được các mục tiêu chính, bao gồm:

- Nghiên cứu nguyên lý hoạt động của YOLOv5: Đã hiểu rõ cách thức thuật toán nhận diện đối tượng.
- Huấn luyện mô hình: Mô hình đã được huấn luyện với hai bộ dữ liệu khác nhau để đánh giá hiệu suất.

- Xây dựng ứng dụng demo: Ứng dụng thực tế được phát triển để minh họa hiệu quả nhận diện đối tượng.

Qua quá trình huấn luyện và thử nghiệm, YOLOv5 đã chứng minh được tiềm năng mạnh mẽ trong việc nhận diện đối tượng thời gian thực, với:

- Độ chính xác cao
- Tốc độ xử lý nhanh

Mô hình đã hoạt động hiệu quả trên bộ dữ liệu lớn và tự tạo, cho phép thực hiện các bài toán nhận diện trong môi trường thực tế dễ dàng.

6.3.2. Hướng phát triển

Nếu có nguồn lực, sẽ có bộ dữ liệu hình ảnh và nhãn cùng các tệp khác đầy đủ hơn, có thể mở rộng ứng dụng của YOLOv5 vào các lĩnh vực như giám sát giao thông, nhận diện khuôn mặt, và phân tích hành vi trong video.

Bên cạnh đó, có thể cải tiến thuật toán bằng cách thử nghiệm với các phiên bản YOLO mới hơn hoặc kết hợp với các kiến trúc học sâu khác như Vision Transformer để nâng cao khả năng nhận diện đối tượng, đặc biệt là các đối tượng nhỏ hoặc bị che khuất. Cũng có thể tối ưu hóa thuật toán để chạy hiệu quả hơn trên các thiết bị có tài nguyên hạn chế như điện thoại di động, giúp mở rộng khả năng ứng dụng của YOLOv5 trong các nền tảng thương mại và các hệ thống độc lập.

6.3.3. Hạn chế của đề tài

Một trong những hạn chế lớn của đề tài là liên quan đến dữ liệu huấn luyện. Mặc dù các bộ dữ liệu từ GitHub và bộ dữ liệu tự tạo đã được sử dụng và gắn nhãn, nhưng các bộ dữ liệu này thiếu tính đa dạng và phong phú. Việc thiếu hình ảnh công khai từ các chủ sở hữu dữ liệu thực tế cùng với sự giới hạn về quy mô của các bộ dữ liệu này có thể ảnh hưởng đến độ chính xác của mô hình khi áp dụng vào các môi trường thực tế phức tạp.

Cuối cùng, một hạn chế quan trọng nữa là về tài nguyên, nguồn lực để thực hiện đề tài. Việc huấn luyện YOLOv5 đòi hỏi tài nguyên tính toán lớn nên cần nguồn lực để tạo dữ liệu và ngoài ra phải cần máy mạnh để khi train dữ liệu có thể diễn ra nhanh hơn.

TÀI LIỆU THAM KHẢO

1. Viblo. (n.d.). Tổng hợp kiến thức từ YOLOv1 đến YOLOv5 (Phần 3). Retrieved from <https://viblo.asia/p/tong-hop-kien-thuc-tu-yolov1-den-yolov5-phan-3-63vKjgJ6Z2R>
2. Ultralytics. (n.d.). YOLOv5 Tutorials: Architecture Description. Retrieved from https://docs.ultralytics.com/vi/yolov5/tutorials/architecture_description/
3. Studocu. (n.d.). Thị giác máy tính: Phát hiện tàu bằng thuật toán YOLOv5. Retrieved from <https://www.studocu.com/vn/document/truong-dai-hoc-fpt/research-project/123doc-thi-giac-may-tinh-phat-hien-tau-bang-thuat-toan-yolov-5/65764493>
4. Shou, J. (n.d.). Thiết kế thuật toán YOLO cho thị giác máy tính. Retrieved from <https://jshou.edu.vn/houjs/article/download/277/242/738>
5. Ultralytics. (n.d.). YOLOv5 Documentation. Retrieved from <https://docs.ultralytics.com/vi/yolov5/>
6. DigitalOcean. (n.d.). How to Train YOLOv5 on Custom Data. Retrieved from <https://www.digitalocean.com/community/tutorials/train-yolov5-custom-data>
7. Cần Thơ University Journal. (n.d.). Ứng dụng YOLO vào thị giác máy tính. Retrieved from <https://ctujsvn.ctu.edu.vn/index.php/ctujsvn/article/download/4014/4291/10999a>
8. Scribd. (n.d.). Chương 3-4-5: Thuật toán YOLO và ứng dụng. Retrieved from <https://www.scribd.com/document/522865591/CH%C6%AF%C6%A0NG-3-4-5>