

TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI

ĐỒ ÁN TỐT NGHIỆP

Nhận diện biển báo và đèn tín hiệu giao thông

NGUYỄN TRƯỜNG GIANG

giang.nt173083@sis.hust.edu.vn

Ngành: Khoa học máy tính

Chuyên ngành: Thị giác Máy tính

Giảng viên hướng dẫn: TS. Nguyễn Hữu Đức

Chữ ký GVHD

Khoa: Khoa học máy tính

Trường: Công nghệ thông tin và Truyền thông

HÀ NỘI, 08/2022

LỜI CAM KẾT

Họ và tên sinh viên: Nguyễn Trường Giang
Điện thoại liên lạc: 0968932100
Email: giang.nt173083@sis.hust.edu.vn
Lớp: KHMT-03
Hệ đào tạo: Kỹ sư chính quy

Tôi – *Nguyễn Trường Giang* – cam kết Đồ án Tốt nghiệp (ĐATN) là công trình nghiên cứu của bản thân tôi dưới sự hướng dẫn của TS. *Nguyễn Hữu Đức*. Các kết quả nêu trong ĐATN là trung thực, là thành quả của riêng tôi, không sao chép theo bất kỳ công trình nào khác. Tất cả những tham khảo trong ĐATN – bao gồm hình ảnh, bảng biểu, số liệu, và các câu trích dẫn – đều được ghi rõ ràng và đầy đủ nguồn gốc trong danh mục tài liệu tham khảo. Tôi xin hoàn toàn chịu trách nhiệm với dù chỉ một sao chép vi phạm quy chế của nhà trường.

Hà Nội, ngày tháng năm

Tác giả ĐATN

LỜI CẢM ƠN

Trước tiên, em xin gửi lời cảm ơn chân thành đến các Thầy cô Trường Công nghệ thông tin và Truyền thông, những người đã đồng hành, truyền đạt cho em những kiến thức cần thiết và bổ ích trong suốt quãng thời gian học Đại học. Những kiến thức quý giá đó đã giúp em xây dựng một nền tảng vững chắc không chỉ áp dụng vào công việc mà còn giúp đỡ em rất nhiều trong cuộc sống. Và đặc biệt em muôn gửi lời cảm ơn đến TS.TS. Nguyễn Hữu Đức, người đã trực tiếp hướng dẫn em trong đồ án tốt nghiệp lần này. Mặc dù chưa có nhiều thời gian làm việc với thầy, nhưng những kiến thức, kinh nghiệm và hướng dẫn của thầy đã giúp em có được định hướng đúng đắn và hoàn thành đồ án một cách hoàn thiện, hiệu quả nhất. Cuối cùng em xin cảm ơn tới gia đình, bạn bè, và các anh chị ở bộ phận IT, công ty TNHH Nippon Steel Metal Products Vietnam, nơi em tham gia thực tập. Trong quãng thời gian làm đồ án có những lúc khó khăn cǎng thẳng, mọi người đã luôn ở bên quan tâm, động viên, giúp đỡ em rất nhiều để em có động lực bước đến ngày hôm nay. Do trong thời gian có hạn và năng lực bản thân còn nhiều hạn chế nên bản báo cáo này không thể tránh khỏi nhiều thiếu sót cá nhân. Em rất mong nhận được những ý kiến đóng góp của các thầy cô để em có thêm kinh nghiệm hoàn thiện kiến thức cho bản thân. Em xin chân thành cảm ơn!

MỤC LỤC

CHƯƠNG 1. GIỚI THIỆU ĐỀ TÀI.....	1
1.1 Đặt vấn đề.....	1
1.2 Các giải pháp hiện tại và hạn chế	3
1.3 Mục tiêu và định hướng giải pháp	4
CHƯƠNG 2. NỀN TẢNG LÝ THUYẾT	5
2.1 Ngữ cảnh của bài toán.....	5
2.1.1 Khái niệm học sâu.....	5
2.1.2 Một số thuật ngữ trong mạng nơ-ron	6
2.1.3 Mạng nơ-ron tích chập – Convolutional neural network (CNN)	11
2.2 Các kết quả nghiên cứu tương tự	12
2.2.1 Mô hình R-CNN	12
2.2.2 Mô hình Fast R-CNN.....	14
2.2.3 Mô hình Faster R-CNN.....	15
2.2.4 Mô hình SSD(Single Shot Detector)	18
2.2.5 Kết luận chương.....	18
CHƯƠNG 3. PHƯƠNG PHÁP ĐỀ XUẤT.....	19
3.1 Bài toán nhận diện biển báo và đèn tín hiệu giao thông	19
3.1.1 Tổng quan bài toán	19
3.1.2 Các điều kiện ràng buộc	19
3.2 Hệ thống phát hiện đối tượng thời gian thực YOLO	19
3.2.1 Tổng quan hệ thống phát hiện đối tượng YOLO.	19
3.2.2 Kiến trúc của mô hình YOLO qua các phiên bản.....	20

CHƯƠNG 4. CÀI ĐẶT CHƯƠNG TRÌNH	25
4.1 Chuẩn bị bộ Dataset.....	25
4.1.1 Thay đổi độ tương phản của ảnh	27
4.1.2 Làm mờ ảnh	29
4.1.3 Loại bỏ phần dữ liệu không đủ điều kiện.....	30
4.1.4 Bổ sung bộ dữ liệu về các phương tiện giao thông để phục vụ cho việc hậu xử lý dữ liệu	31
4.1.5 Cấu trúc bộ dữ liệu phục vụ cho quá trình huấn luyện.....	32
4.2 Thiết lập môi trường huấn luyện nhận dạng biển báo và đèn tín hiệu giao thông	33
4.3 Traning.....	34
4.4 Xử lý các trường hợp nhận diện nhầm đèn tín hiệu với đèn của các phương tiện giao thông	34
4.5 Kết quả chương trình.....	36
CHƯƠNG 5. KẾT LUẬN	41
5.1 Kết luận	41
5.2 Hướng phát triển trong tương lai	41
TÀI LIỆU THAM KHẢO.....	42

DANH MỤC HÌNH VẼ

Hình 2.1	Tế bào sinh học	5
Hình 2.2	Kiến trúc mạng nơ-ron gồm k tầng ẩn	6
Hình 2.3	Một số hàm kích hoạt phi tuyến phổ biến [1]	8
Hình 2.4	Ví dụ về IoU [1]	8
Hình 2.5	Xác định các mỏ neo và hộp dự đoán	9
Hình 2.6	Ví dụ về Upsampling	10
Hình 2.7	Thành phần một mạng nơ-ron tích chập [1]	11
Hình 2.8	Tầng Convolution [1]	11
Hình 2.9	Tầng Pooling [1]	12
Hình 2.10	Tầng Fully Connected [1]	12
Hình 2.11	Các bước thực hiện thuật toán R-CNN [1]	13
Hình 2.12	Đầu vào và đầu ra thuật toán Selective search algorithm [1]	13
Hình 2.13	Các bước xử lý hình ảnh để lấy region proposal [2]	14
Hình 2.14	Hình ảnh minh họa thuật toán Fast R - CNN[2]	15
Hình 2.15	Kiến trúc của mạng Faster R-CNN[2]	16
Hình 2.16	Cách lấy toạ độ một bức ảnh	16
Hình 2.17	Xác định tâm của bức ảnh bằng RPN [2]	17
Hình 2.18	Mô hình mạng SSD [2]	18
Hình 3.1	YOLO version 1 [3]	20
Hình 3.2	Loại bỏ phần kết nối đầy đủ trên YOLOv2 [3]	20
Hình 3.3	Kết quả dự đoán của YOLO[3]	21
Hình 3.4	Mô hình YOLO version 3	21
Hình 3.5	Kết quả dự đoán của YOLOv3	22
Hình 3.6	Cấu trúc nhận diện vật thể của YOLOv5[3]	23
Hình 4.1	Bộ dữ liệu thu thập được	25
Hình 4.2	Kết quả thu được ở điều kiện tốt	26
Hình 4.3	Kết quả thu được ở điều kiện sương mù	26
Hình 4.4	Một ảnh trong bộ dữ liệu được chụp dưới điều kiện ánh sáng tốt	28
Hình 4.5	Bức ảnh sau khi tăng độ tương phản	29
Hình 4.6	Bức ảnh sau khi giảm độ tương phản	29
Hình 4.7	Tập ảnh với các điều kiện sáng khác nhau	29
Hình 4.8	Ảnh gốc khi chưa chạy qua bộ lọc làm mờ	30
Hình 4.9	Ảnh gốc sau khi chạy qua bộ lọc làm mờ	30
Hình 4.10	Bộ ảnh với các độ blur khác nhau tạo ra từ ảnh gốc	30

Hình 4.11	Dễ dàng nhâm lẩn đèn từ các phương tiện giao thông thành đèn tín hiệu giao thông	31
Hình 4.12	Bộ dữ liệu bổ sung cho việc nhận diện phương tiện giao thông, phục vụ hậu xử lý sau này	32
Hình 4.13	Cấu trúc thư mục train_data	32
Hình 4.14	Gán nhãn hình ảnh bằng LabelImg	33
Hình 4.15	Clone model và cài đặt các requirements	33
Hình 4.16	Cấu hình lại file yaml	34
Hình 4.17	Model nhận diện ô tô và đèn	35
Hình 4.18	Confusion_matrix	37
Hình 4.19	Sơ đồ P_curve	37
Hình 4.20	Sơ đồ R_curve	38
Hình 4.21	Kết quả huấn luyện	38
Hình 4.22	Phát hiện biển báo Cấm dừng, đỗ	39
Hình 4.23	Phát hiện đèn xanh, biển cấm rẽ, biển cấm ngược chiều	39
Hình 4.24	Phát hiện đèn đỏ và biển báo Cấm đi ngược chiều	39
Hình 4.25	Phát hiện biển báo Cấm dừng, đỗ và cấm đi ngược chiều trong điều kiện ban đêm	40
Hình 4.26	Phát hiện đèn tín hiệu trong điều kiện ban đêm	40
Hình 4.27	Phát hiện đèn tín hiệu trong điều kiện nắng gắt	40

DANH MỤC THUẬT NGỮ VÀ TỪ VIẾT TẮT

Thuật ngữ	Ý nghĩa
AI	Trí tuệ nhân tạo (Artificial Intelligence)
Anchors box	Hộp mỏ neo
Bounding box	Hộp dự đoán
Classification loss	Lớp tính toán sự mất mát do gán nhãn
CNN	Mạng nơ-ron tích chập (Convolutional Neural Networks)
Confidence loss	Lớp tính toán sự mất mát do lấy hộp dự đoán
CONV	Lớp tích chập trong mạng nơ-ron (Convolutional)
FC	Lớp đầy đủ trong mạng nơ-ron (Fully connected)
IoU	tỉ lệ trùng vào nhau của 2 box(Intersection Over Union)
Label	nhãn của ảnh
Localization loss	Lớp tính toán sự mất mát do lấy toạ độ tâm
Non-max suppression	Phương pháp giảm số khung hình dự đoán
POOL	Lớp trích đặc trưng và giảm chiều trong mạng nơ-ron (Pooling)
R - CNN	Region with Convolutional Neural Networks
RNN	Mạng nơ-ron hồi quy (Recurrent Neural Networks)
Training Data	Dữ liệu huấn luyện
Upsampling	Hàm tăng kích thước ảnh
YOLO	Thuật toán object detection có mục tiêu dự báo nhãn cho vật thể và xác định location của vật thể (You Only Look Once)

Thuật ngữ | **Ý nghĩa**

CHƯƠNG 1. GIỚI THIỆU ĐỀ TÀI

1.1 Đặt vấn đề

Bài toán nhận diện vật thể ngày càng phổ biến trong đời sống của chúng ta. Hơn một thập kỷ qua, có rất nhiều công trình nghiên cứu về bài toán xác định vật thể. Các nghiên cứu đi từ đơn giản là bức ảnh chỉ có một vật, rồi một vật với số lượng nhiều trong một ảnh, và sau đó là nhiều vật trong cùng một bức ảnh. Bên cạnh việc nhận diện, phát hiện được vị trí của vật thể, một công việc khó khăn hơn chính là làm sao để xác định được chi tiết của từng đối tượng đến từng điểm ảnh trong bức ảnh. Cùng với sự chính xác trong việc nhận diện chính là tốc độ cần đủ để nhận diện vật thể trong thời gian thực đáp ứng nhu cầu thực tiễn cần thiết cho con người.

Computer vision có khá nhiều các nhiệm vụ khác nhau và đôi khi ta gặp khó khăn trong việc phân biệt các nhiệm vụ này. Chẳng hạn như phân loại hình ảnh (image classification) là gì? Định vị vật thể (object localization) là gì? Sự khác biệt giữa định vị vật thể (object localization) và phát hiện đối tượng (object detection) là gì? Đây là những khái niệm có thể gây nhầm lẫn, đặc biệt là khi cả ba nhiệm vụ đều liên quan đến nhau. Hiểu một cách đơn giản:

- Phân loại hình ảnh (image classification): liên quan đến việc gán nhãn cho hình ảnh.
- Định vị vật thể (object localization): liên quan đến việc vẽ một hộp giới hạn (bounding box) xung quanh một hoặc nhiều đối tượng trong hình ảnh nhằm khoanh vùng đối tượng.
- Phát hiện đối tượng (object detection): Là nhiệm vụ khó khăn hơn và là sự kết hợp của cả hai nhiệm vụ trên: Vẽ một bounding box xung quanh từng đối tượng quan tâm trong ảnh và gán cho chúng một nhãn. Kết hợp cùng nhau, tất cả các vấn đề này được gọi là object recognition hoặc object detection.

Phân loại hình ảnh liên quan đến việc dự đoán lớp của một đối tượng trong một hình ảnh. Định vị vật thể đề cập đến việc xác định vị trí của một hoặc nhiều đối tượng trong một hình ảnh và vẽ bounding box xung quanh chúng. Phát hiện đối tượng kết hợp hai nhiệm vụ trên và thực hiện cho một hoặc nhiều đối tượng trong hình ảnh. Chúng ta có thể phân biệt giữa ba nhiệm vụ thị giác máy tính cơ bản trên thông qua input và output của chúng như sau:

- Phân loại hình ảnh: Dự đoán nhãn của một đối tượng trong một hình ảnh
 - Input: Một hình ảnh với một đối tượng, chẳng hạn như một bức ảnh.

- Output: Nhãn lớp (ví dụ: một hoặc nhiều số nguyên được ánh xạ tới nhãn lớp).
- Định vị đối tượng: Xác định vị trí hiện diện của các đối tượng trong ảnh và cho biết vị trí của chúng bằng bounding box.
 - Input: Một hình ảnh có một hoặc nhiều đối tượng, chẳng hạn như một bức ảnh.
 - Output: Một hoặc nhiều bounding box được xác định bởi tọa độ tâm, chiều rộng và chiều cao.
- Phát hiện đối tượng: Xác định vị trí hiện diện của các đối tượng trong bounding box và nhãn của các đối tượng nằm trong một hình ảnh.
 - Input: Một hình ảnh có một hoặc nhiều đối tượng, chẳng hạn như một bức ảnh.
 - Một hoặc nhiều bounding box và nhãn cho mỗi bounding box.

Bài toán phát hiện đối tượng đóng vai trò quan trọng trong các hệ thống phát hiện các khối u trong cơ thể người hay phát hiện làn đường , các vật thể cản trở giao thông, biển báo, đèn tín hiệu giao thông ... Bài toán phát hiện đối tượng giúp ích rất nhiều trong cuộc sống.

NogiCargo là một phần mềm hỗ trợ cho việc điều khiển xe tự động do đội ngũ bộ phận IT công ty NSMV xây dựng và phát triển. Phần mềm cung cấp các tính năng để tự hành cho ô tô mà không cần đến sự tham gia của con người. Những tính năng chính mà phần mềm cung cấp có thể kể đến như: sử dụng bản đồ và GPS để xác định cung đường đi cho thuận tiện, nhận dạng các biển báo và đèn tín hiệu giao thông, phát hiện và theo dõi các phương tiện, vật cản xuất hiện trên đường, nhận diện vạch kẻ, làn đường, ... Và trong quá trình thực tập tại công ty, em đã vinh dự được tham gia vào quá trình nghiên cứu và phát triển phần mềm, cụ thể là xây dựng hệ thống nhận diện đèn tín hiệu và biển báo giao thông. Sau quá trình tham gia dự án cũng như được sự hỗ trợ, giúp đỡ của các anh chị trong công ty, hệ thống của em cũng đã có những kết quả khả quan. Vì vậy trong đồ án lần này em xin phép được trình bày những công việc và kết quả em đã tìm hiểu và xây dựng được trong thời gian tham gia xây dựng phần mềm NogiCargo.

Giới thiệu qua về tầm quan trọng của hệ thống, trong thời đại công nghệ phát triển, tai nạn giao thông vẫn là một vấn đề được quan tâm thường ngày. Để giúp cho việc tham gia giao thông an toàn hơn, giảm thiểu các vụ tai nạn giao thông đáng tiếc xảy ra cần một hệ thống hỗ trợ phát hiện vật cản cũng như phát hiện biển báo, đèn tín hiệu. Hệ thống sẽ giúp cho việc xác định nguy hiểm khi gặp vật cản

cũng như giúp cho người tham gia đi đúng luật. Nếu chỉ sử dụng các hộp giới hạn sẽ không thể xác định được độ lớn của vật cản cũng như độ lớn của biển báo và đèn tín hiệu giao thông, điều này cũng sẽ không giúp ích nhiều trong việc xác định và đưa ra các quyết định kịp thời khi tham gia giao thông. Nhận thấy được tầm quan trọng của các vấn đề trên, phần mềm NogiCargo đã xây dựng một tính năng là nhận diện đèn tín hiệu và biển báo giao thông nhằm hỗ trợ cho việc điều khiển tự động trên ô tô tuân thủ đúng luật lệ và an toàn hơn trong quá trình tham gia giao thông.

1.2 Các giải pháp hiện tại và hạn chế

Phát hiện biển báo và đèn tín hiệu giao thông là một trong những vấn đề rất quan trọng trong nhiều ứng dụng về các hệ thống hỗ trợ giao thông tự động, đặc biệt trong các hệ thống điều khiển xe tự động. Trong khoảng từ năm 1998 đến nay, cùng với sự phát triển vượt bậc về tốc độ xử lý của máy tính, giá thành của các thiết bị hỗ trợ ngày càng giảm thì các nghiên cứu về hệ thống điều khiển xe tự động ngày càng được phát triển. Đã có nhiều nghiên cứu tập trung vào bài toán phát hiện biển báo, đèn tín hiệu giao thông và một số kết quả bước đầu đạt được đã cho thấy những tín hiệu khả quan. Trong bài toán phát hiện biển báo và đèn tín hiệu giao thông có 2 loại mục tiêu chính là:

- Nhận dạng các loại đèn tín hiệu và một số loại biển báo giao thông phổ biến
- Đưa ra thông báo để hỗ trợ con người trong quá trình lái xe, điều khiển xe tự động

Trong đó, mục tiêu phục vụ cho hệ thống điều khiển xe tự động được đánh giá là khó nhất. Có khá nhiều nghiên cứu trong thời gian gần đây tập trung ở mục tiêu hỗ trợ hệ thống điều khiển xe tự động, tuy vậy vẫn còn nhiều khó khăn tồn tại trong việc giải quyết bài toán với mục tiêu này nói riêng cũng như toàn bộ bài toán phát hiện biển báo và đèn tín hiệu giao thông nói chung. Các khó khăn có thể chỉ ra như sau:

- Phần lớn yêu cầu đòi hỏi ứng dụng phải xử lý ở thời gian thực. Trong rất nhiều nghiên cứu chỉ ra các kết quả đạt được khá tốt, tuy nhiên các nghiên cứu này vẫn chưa thực thi được trong thời gian thực
- Khó khăn nhận diện chính xác khi vị trí các đối tượng bị che khuất
- Sự phức tạp của địa hình, thời tiết, các vật thể gây nhiễu...

Hiện nay, có rất nhiều hệ thống phát hiện biển báo và đèn tín hiệu giao thông với độ chính xác cao, điển hình là hệ thống phát hiện biển báo và đèn tín hiệu giao thông sử dụng mô hình Mask R-CNN. Tuy nhiên tốc độ xử lý của mô hình này lại rất chậm, chỉ khoảng 7 khung hình/ giây. Điều này là một cản trở rất lớn trong việc

áp dụng vào những bài toán phát hiện biển báo và đèn tín hiệu giao thông thực tế yêu cầu tốc độ xử lý phải trên 30 khung hình/giây.

1.3 Mục tiêu và định hướng giải pháp

Trong phần mềm NogiCargo, chúng ta sẽ tập trung vào vấn đề xử lý thời gian. Vậy làm sao để tốc độ xử lý đạt được thời gian thực?

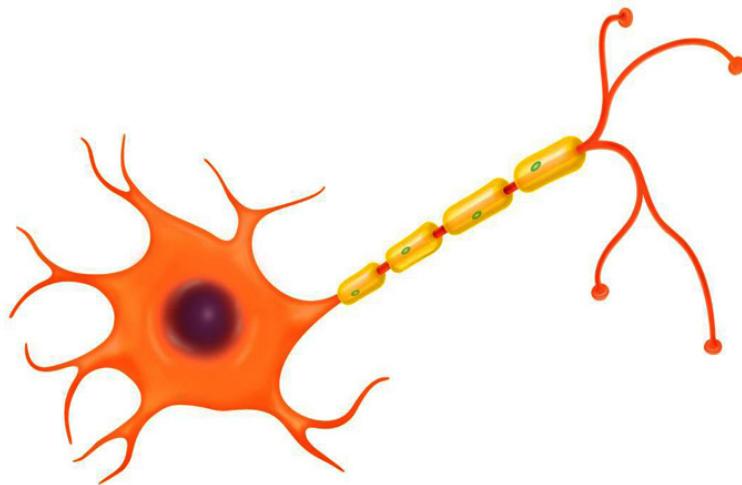
Vấn đề mà các mô hình trước đó gặp phải là tính ‘tuần tự’ trong các bước tính toán. Để giải quyết vấn đề này, chúng ta phải tìm hiểu một thuật toán có các bước tính toán được thực hiện song song nhau, và bỏ qua hoặc thay đổi những bước không quan trọng, có thể thay đổi để giảm bớt thời gian thực thi tính toán trong mô hình. YOLO là một phương pháp phát hiện đối tượng thời gian thực hiện đại nhất hiện nay. Với GPU Titan X mô hình YOLO có tốc độ 30 FPS với độ chính xác trung bình 57.9% trên tập dữ liệu tiêu chuẩn COCO. YOLO rất phù hợp cho các bài toán đòi hỏi tốc độ xử lý theo thời gian thực như giám sát giao thông, xe tự lái, các hệ thống giám sát an ninh. Vì vậy hệ thống đã áp dụng thuật toán YOLO vào việc nhận dạng đèn tín hiệu và biển báo giao thông.

CHƯƠNG 2. NỀN TẢNG LÝ THUYẾT

2.1 Ngữ cảnh của bài toán

2.1.1 Khái niệm học sâu

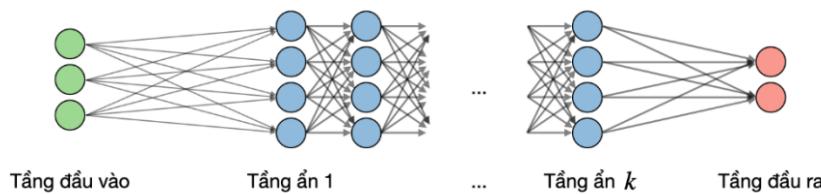
Học sâu (deep learning) là một nhánh của ngành máy học, dựa trên một tập hợp các thuật toán để cố gắng mô hình dữ liệu để trừu tượng hóa ở mức cao bằng cách sử dụng nhiều lớp xử lý với cấu trúc phức tạp, hoặc bằng cách khác bao gồm nhiều lớp biến đổi phi tuyến để trích tách đặc trưng và chuyển đổi. Mỗi lớp kế tiếp dùng đầu ra của lớp trước làm đầu vào. Các thuật toán này có thể được giám sát hoặc không cần giám sát và các ứng dụng bao gồm các mô hình phân tích (không giám sát) và phân loại (giám sát). Một trong những phương pháp học sâu thành công nhất là mô hình mạng nơ-ron nhân tạo (Artificial Neural Network). Mạng nơ-ron nhân tạo được lấy cảm hứng từ các mô hình sinh học năm 1959 được đề xuất bởi người đoạt giải Nobel David H. Hubel Torsten Wiesel, 2 người đã tìm thấy hai loại tế bào trong vỏ não thị giác chính: các tế bào đơn giản và các tế bào phức tạp. Nhiều mạng nơ-ron nhân tạo có thể được xem như là các mô hình ghép tầng của các tế bào loại lấy cảm hứng từ những quan sát sinh học.



Hình 2.1: Tế bào sinh học

Mạng neural học sâu là một mô hình mạng neural có nhiều tầng kết hợp với nhau. Mỗi tầng mạng là đầu ra của tầng trước và là đầu vào của tầng sau.

Độ sâu của mô hình được tính bằng số tầng ẩn cộng một.



Hình 2.2: Kiến trúc mạng nơ-ron gồm k tầng ẩn

Kiến trúc chung của mạng nơ-ron nhân tạo bao gồm thành phần: Tầng đầu vào, Tầng ẩn và Tầng đầu ra.

- **Tầng đầu vào (Input layer):** Là nơi để nạp dữ liệu vào. Mỗi nơ-ron tương ứng với một thuộc tính (attribute) hoặc đặc trưng (feature) của dữ liệu đầu vào.
- **Tầng ẩn (Hidden layer):** Là nơi xử lý dữ liệu trước khi đưa ra output. Thường là hàm tổng (Summation function) để đưa ra giá trị của một nơ-ron tại hidden layer. Có thể có nhiều hơn một hidden layer. Khi đó đầu ra của một hidden layer sẽ là đầu vào cho hidden layer tiếp theo.
- **Lớp đầu ra (Output layer):** Là giá trị đầu ra của mạng nơ-ron. Sau khi đi qua hidden layer cuối cùng, dữ liệu sẽ được chuyển hóa bằng một hàm số được gọi là hàm kích hoạt (Activation function) và đưa ra output cuối cùng. Hàm chuyển đổi thường là hàm tanh(x), sigmoid(x) hoặc softmax(x).

Cập nhật trọng số: Trong mạng nơ-ron, trọng số được cập nhật như sau:

- Bước 1: Lấy một batch dữ liệu huấn luyện.
- Bước 2: Thực hiện việc lan truyền tiến để lấy được lỗi tương ứng.
- Bước 3: Lan truyền ngược lỗi để lấy được gradients.
- Bước 4: Sử dụng gradients để cập nhật trọng số của mạng.

2.1.2 Một số thuật ngữ trong mạng nơ-ron

Hàm softmax: Bước softmax có thể được coi là một hàm logistic tổng quát lấy đầu vào là một vec-tơ chứa các giá trị $x \in R^n$ và cho ra là một vec-tơ gồm các xác suất $p \in R^n$ thông qua một hàm softmax ở cuối kiến trúc. Nó được định nghĩa như sau:

$$p = \begin{pmatrix} p_1 \\ \vdots \\ p_n \end{pmatrix} \quad \text{với} \quad p_i = \frac{e^{x_i}}{\sum_{j=1}^n e^{x_j}}$$

Hàm lỗi (Cross-Entropy): Trong bối cảnh của mạng neural, hàm lỗi cross-entropy $L(z,y)$ thường được sử dụng và định nghĩa như sau:

$$L(z, y) = -[y \log(z) + (1 - y) \log(1 - z)] [1]$$

Hàm kết hợp Softmax CrossEntropy : là sự kết hợp của lớp fully connected cuối cùng.

Tốc độ học (Learning rate): tốc độ học - thường được ký hiệu bởi α hoặc đôi khi là η , chỉ ra tốc độ mà trọng số được cập nhật. Thông số này có thể là cố định hoặc được thay đổi tùy biến. Phương thức phổ biến nhất hiện tại là Adam, trong đó phương thức thay đổi tốc độ học một cách phụ hợp nhất có thể.

Hàm kích hoạt (Activation function): Hàm kích hoạt được sử dụng ở phần cuối của đơn vị ẩn để đưa ra độ phức tạp phi tuyến tính cho mô hình và được sử dụng làm dữ liệu đầu vào cho tầng tiếp theo.

Hàm kích hoạt có 3 loại:

- Hàm kích hoạt tuyến tính (Linear Activation Function)
- Hàm kích hoạt nhị phân (Binary Step Function)
- Hàm kích hoạt phi tuyến (non-Linear Activation Function)

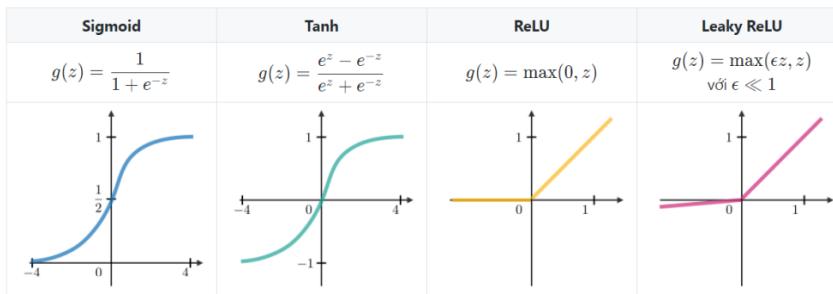
Hàm kích hoạt tuyến tính: là một hàm có dạng $f(x) = cx$. hàm này sử dụng tích ô giữa đầu vào và trọng số của mỗi nơ-ron và tạo ra một tín hiệu đầu ra tỉ lệ với đầu vào. Nhược điểm của hàm tuyến tính gồm 2 vấn đề:

- Không thể sử dụng lan truyền ngược để đào tạo mô hình vì đạo hàm của hàm này là một hằng số nên không thể có mối liên hệ với đầu vào.
- Nó chỉ có chức năng như một hàm hồi quy tuyến tính, vì vậy sẽ hạn chế trong việc tính toán và xử lý các thông số phức tạp của dữ liệu đầu vào.

Hàm kích hoạt nhị phân: là một hàm kích hoạt dựa trên ngưỡng. Nếu một giá trị đầu vào cao hơn hoặc thấp hơn một ngưỡng cho trước, mạng nơ-ron sẽ được kích hoạt và gửi tín hiệu chính xác đến lớp tiếp theo. Đầu ra của hàm kích hoạt nhị phân chỉ có hai giá trị. Vì vậy trong một số bài toán đa chiều thì hàm này không thể giải quyết được.

Hàm kích hoạt phi tuyến: là một hàm cho phép mô hình tạo ra ánh xạ giữa đầu vào và đầu ra của mạng. Điều này phù hợp cho việc đào tạo các mô hình phức tạp của các tập dữ liệu phi tuyến có kích thước lớn. Phù hợp sử dụng lan truyền ngược để cập nhật các trọng số của mô hình.

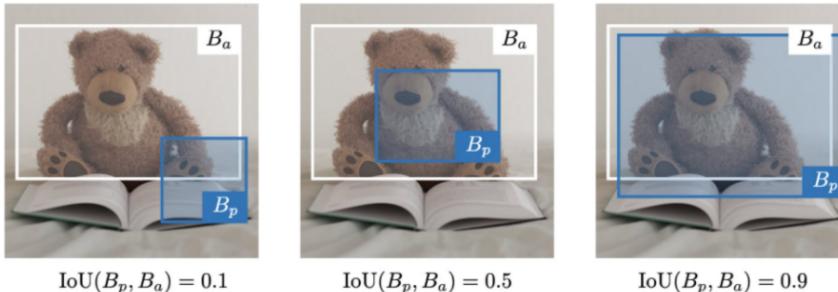
Dưới đây là một số hàm kích hoạt phi tuyến thường dùng:

**Hình 2.3:** Một số hàm kích hoạt phi tuyến phổ biến [1]

Tỷ lệ IoU (Intersection over Union): Tỷ lệ vùng giao trên vùng hợp, là một hàm định lượng vị trí β_p của một hộp giới hạn dự đoán được định vị đúng như thế nào so với hộp giới hạn thực tế β_a . Nó được định nghĩa:

$$\text{IoU}(B_p, B_a) = \frac{B_p \cap B_a}{B_p \cup B_a}$$

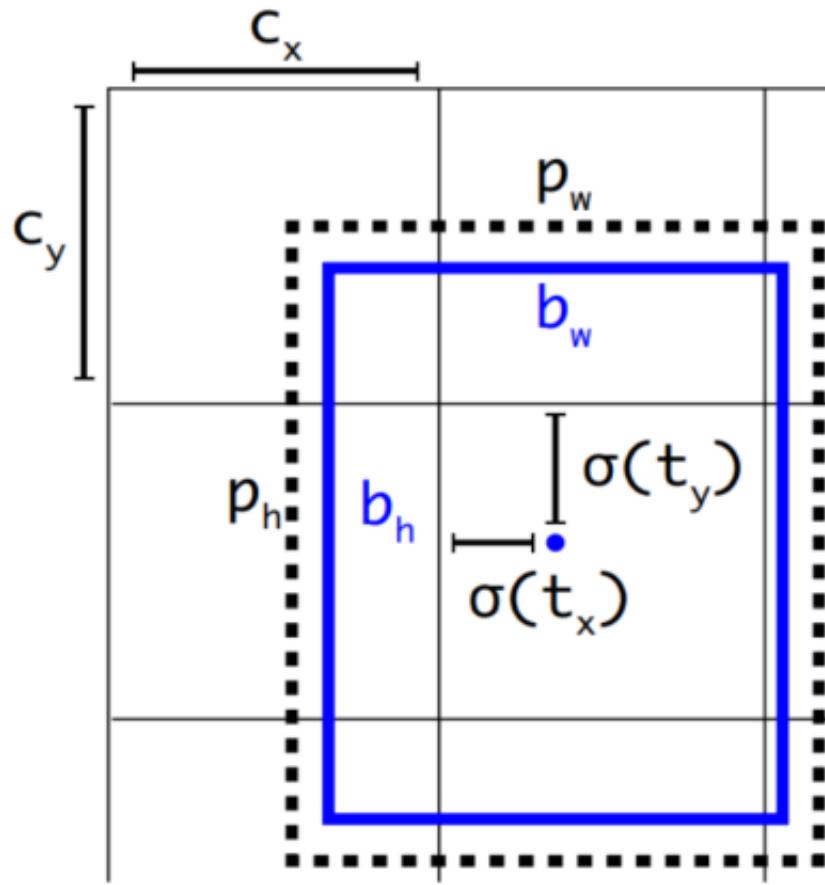
Lưu ý: ta luôn có IoU nằm trong khoảng [0,1]. Và hộp giới hạn β_p được cho là khá tốt nếu $\text{IoU}(\beta_p, \beta_a) \geq 0.5$.

**Hình 2.4:** Ví dụ về IoU [1]

Hộp mỏ neo (Anchor boxes): đây là một kỹ thuật được dùng để dự đoán những hộp giới hạn nằm chồng lên nhau. Trong thực nghiệm, mạng được phép dự đoán nhiều hơn một hộp giới hạn cùng một lúc, trong đó mỗi dự đoán được giới hạn theo một tập những tính chất hình học cho trước. Ví dụ, dự đoán đầu tiên là một hình chữ nhật ngang, và dự đoán thứ hai có thể là một hình chữ nhật đứng hay một hình có kích thước hình học khác.

Anchors là sự sắp xếp độ ưu tiên bouding box, mà chúng ta tính toán trên tập dữ liệu sử dụng k-mean. Chúng ta sẽ dự đoán độ rộng và chiều cao của hộp từ cụm trung tâm. Tọa độ điểm trung tâm của hộp liên quan tới vị trí của phần lọc đã được dự đoán sử dụng hàm sigmoid.

Theo công thức mô tả các đầu ra mạng được biến đổi để bắt buộc dự đoán bounding box:



Hình 2.5: Xác định các mỏ neo và hộp dự đoán

Ở đây b_x, b_y, b_w, b_h là tọa độ trung tâm x,y, chiều rộng và chiều cao của hộp dự đoán. t_x, t_y, t_w, t_h là đầu ra của mạng. c_x và c_y là tọa độ của lưới ô. p_w và p_h là chiều của hộp. Công thức tính các giá trị trên:

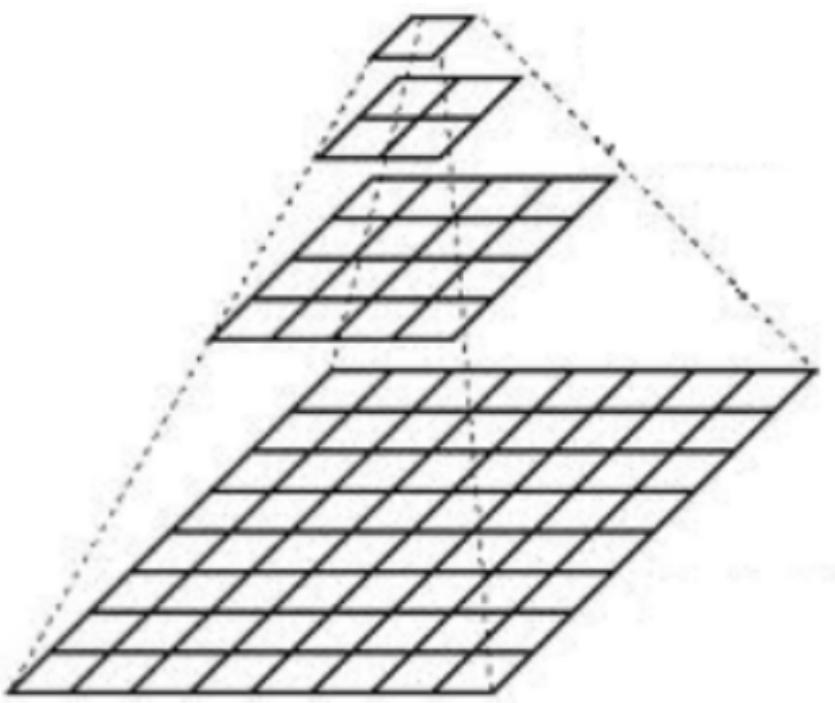
$$\begin{aligned} b_x &= \sigma(t_x) + c_x \\ b_y &= \sigma(t_y) + c_y \\ b_w &= p_w e^{t_w} \\ b_h &= p_h e^{t_h} \end{aligned}$$

Tọa độ trung tâm Chúng ta dự đoán tọa độ trung tâm thông qua hàm sigmoid. Điều này bắt buộc giá trị của đầu ra phải nằm giữa [0, 1].

Non-max suppression: Đây là một kỹ thuật hướng tới việc loại bỏ những hộp giới hạn bị trùng lặp được chồng lên nhau của cùng một đối tượng bằng cách chọn hộp chứa nhiều đặc trưng nhất và loại bỏ những hộp giới hạn khác.

Upsampling: là tích chập ngược. Một cách khác để kết nối đầu ra thô với các pixel dày đặc là nội suy. Ví dụ, phép nội suy song tuyến tính đơn giản tính toán

từng đầu ra từ bốn đầu vào gần nhất bằng một bản đồ tuyến tính chỉ phụ thuộc vào vị trí tương đối của các ô đầu vào và đầu ra. Theo một nghĩa nào đó, upampling với yếu tố f là tích chập với bước tiến đầu vào phân đoạn là $1/f$. Vì vậy, miến f là tích phân, do đó, một cách tự nhiên để lấy mẫu là tích chập ngược (đôi khi được gọi là giải mã) với bước tiến đầu ra là f . Một hoạt động như vậy là không đáng kể để thực hiện, vì nó chỉ đơn giản là đảo ngược các bước tiến và lùi của tích chập. Do đó, việc lấy mẫu được thực hiện trong mạng để học từ đầu đến cuối bằng cách truyền ngược từ mất pixel.



Hình 2.6: Ví dụ về Upsampling

Mục đích của hàm này là tăng kích thước cho bức ảnh nhằm giảm mất mát và tính gần đúng các giá trị pixel bị thiếu.

Các bước thực hiện:

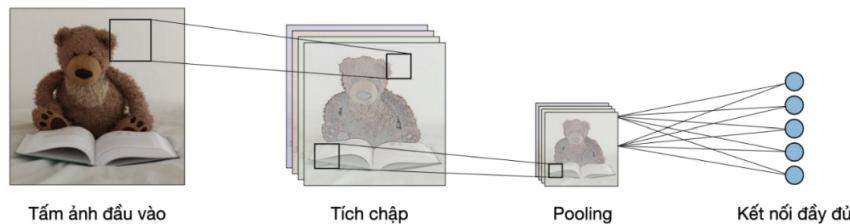
- Đầu tiên, tăng kích thước hình ảnh lên gấp đôi hình ảnh gốc ở mỗi chiều, với các hàng chẵn mới
- Thực hiện phép chập với cùng một bộ số nhân được hiển thị ở trên để tính gần đúng giá trị của "pixel bị thiếu"

Upsampling dựa trên lý thuyết về hình ảnh của kim tự tháp. Tức là một tập hợp các hình ảnh xuất phát từ một hình ảnh gốc được lấy mẫu liên tiếp cho đến khi đạt được một số điểm dừng mong muốn. Phương pháp upsampling dựa trên loại hình Kim tự tháp Laplacian - Được sử dụng để tái tạo một hình ảnh được lấy mẫu từ một

hình ảnh nhỏ hơn trong kim tự tháp.

2.1.3 Mạng nơ-ron tích chập – Convolutional neural network (CNN)

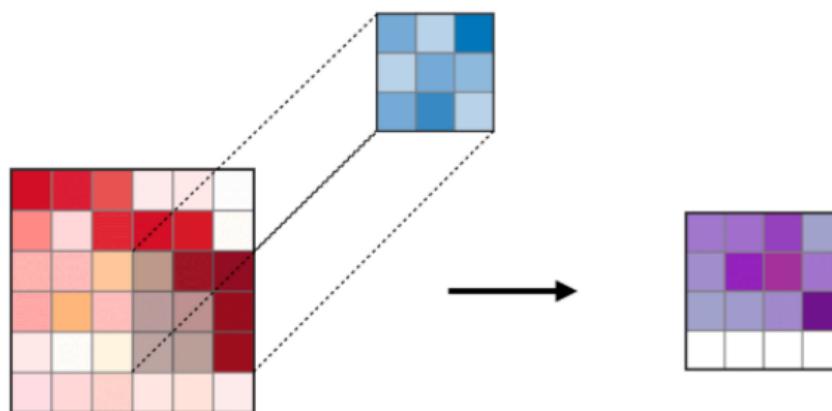
Mạng neural tích chập là mô hình mạng neural hoạt động hiệu quả dựa trên dũa liệu ảnh nhờ khả năng nhận diện tốt các đường nét đặc trưng của đối tượng trong ảnh.



Hình 2.7: Thành phần một mạng nơ-ron tích chập [1]

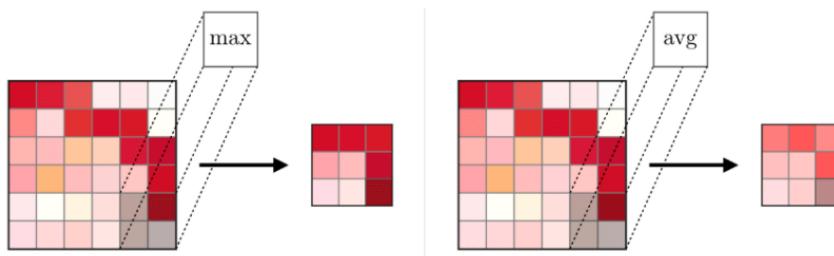
Mạng Neural tích chập gồm 3 tầng chính:

- Tầng tích chập (CONV): sử dụng các bộ lọc để thực hiện phép tích chập khi đưa chúng đi qua đầu vào I theo các chiều của nó. Các siêu tham số của các bộ lọc này bao gồm kích thước bộ lọc F và độ trượt (stride) S. Kết quả đầu ra O được gọi là feature map hay activation map.

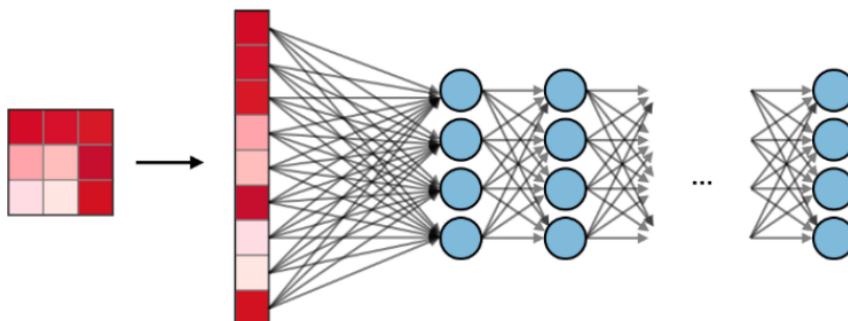


Hình 2.8: Tầng Convolution [1]

- Tầng Pooling (POOL): là một phép downsampling, thường được sử dụng sau tầng tích chập, giúp tăng tính bất biến không gian. Cụ thể, max pooling và average pooling là những dạng pooling đặc biệt, mà tương ứng là trong đó giá trị lớn nhất và giá trị trung bình được lấy ra.

**Hình 2.9:** Tầng Pooling [1]

- Tầng Fully Connected (FC): nhận đầu vào là các dữ liệu đã được làm phẳng, mà mỗi đầu vào đó được kết nối đến tất cả neuron. Trong mô hình mạng CNNs, các tầng kết nối đầy đủ thường được tìm thấy ở cuối mạng và được dùng để tối ưu hóa mục tiêu của mạng ví dụ như độ chính xác của lớp.

**Hình 2.10:** Tầng Fully Connected [1]

2.2 Các kết quả nghiên cứu tương tự

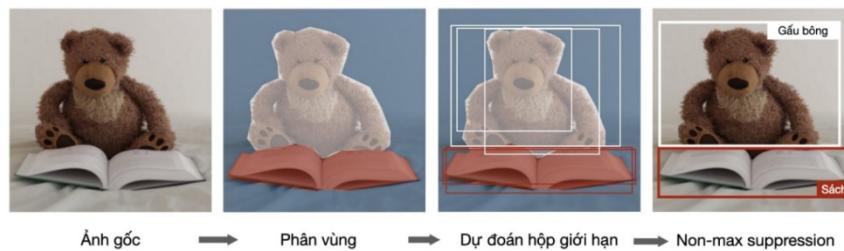
Trong phần này em xin phép trình bày về một số thuật toán phổ biến liên quan đến bài toán phát hiện đối tượng và những hạn chế của chúng.

2.2.1 Mô hình R-CNN

R-CNN là một thuật toán phát hiện vật thể mà đầu tiên phân chia ảnh thành các vùng để tìm các hộp giới hạn có khả năng liên quan cao rồi chạy một thuật toán phát hiện để tìm những thứ có khả năng cao là vật thể trong những hộp giới hạn đó.

Ý tưởng thuật toán R-CNN khá đơn giản, gồm 2 bước:

- Bước 1: Dùng Selective Search algorithm để lấy ra khoảng 2000 bounding box trong input mà có khả năng chứa đối tượng.
- Bước 2: Với mỗi bounding box ta xác định xem nó là đối tượng nào (người, ô tô, xe đạp,...)



Hình 2.11: Các bước thực hiện thuật toán R-CNN [1]

Selective search algorithm: Input của thuật toán là ảnh màu, output là khoảng 2000 bounding box mà có khả năng chứa các đối tượng.



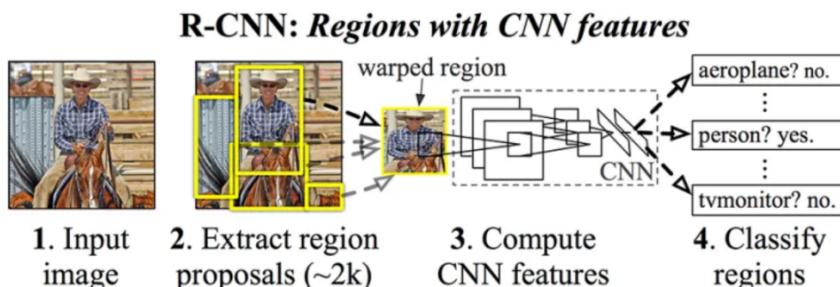
Hình 2.12: Đầu vào và đầu ra thuật toán Selective search algorithm [1]

Nhận xét: Ta không thể dùng mỗi màu trong output để làm 1 region proposal được vì:

- Mỗi đối tượng trong ảnh có thể chứa nhiều hơn 1 màu.
- Các đối tượng bị che mất một phần như cái đĩa dưới cái chén không thể xác định được.

→ Cần nhóm các vùng màu với nhau để làm region proposal.

Phân loại region proposal: Bài toán trở thành phân loại ảnh cho các region proposal. Do thuật toán selective search cho tới 2000 region proposal nên có rất nhiều region proposal không chứa đối tượng nào. Vậy nên ta cần thêm 1 lớp hình ảnh nền (không chứa đối tượng nào). Ví dụ như hình dưới ta có 4 region proposal, ta sẽ phân loại mỗi bounding box là người, ngựa hay hình ảnh nền.



Hình 2.13: Các bước xử lý hình ảnh để lấy region proposal [2]

Sau đó các region proposal được resize lại về cùng kích thước và thực hiện transfer learning với feature extractor, sau đó các extracted feature được cho vào thuật toán SVM để phân loại ảnh.

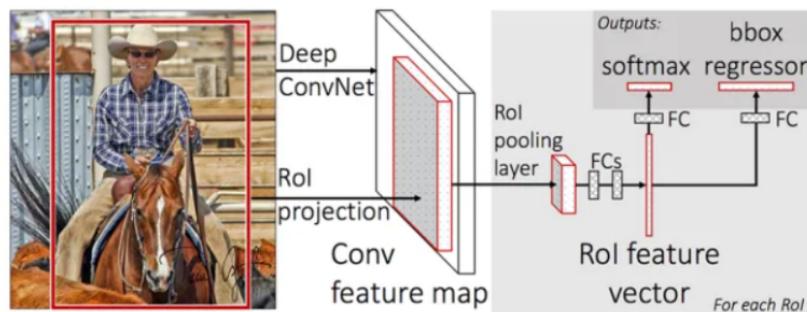
Bên cạnh đó thì extracted feature cũng được dùng để dự đoán 4 offset values cho mỗi cạnh. Ví dụ như khi region proposal chứa người nhưng chỉ có phần thân và nửa mặt, nửa mặt còn lại không có trong region proposal đó thì offset value có thể giúp mở rộng region proposal để lấy được toàn bộ người.

Đánh giá R - CNN: Hồi mới xuất hiện thì thuật toán hoạt động khá tốt cho với các thuật toán về computer vision trước đó nhờ vào CNN, tuy nhiên nó vẫn có khá nhiều hạn chế:

- Vì với mỗi ảnh ta cần phân loại các class cho 2000 region proposal nên thời gian train rất lâu.
- Không thể áp dụng cho real-time vì mỗi ảnh trong test set mất tới 47s để xử lý.

2.2.2 Mô hình Fast R-CNN

Fast R-CNN được giới thiệu vào năm 2015. Fast R -CNN giải quyết được một số hạn chế của R-CNN để cải thiện tốc độ. Bằng việc xử lý tất cả các vùng được đề xuất cùng nhau trong CNN bằng cách sử dụng lớp ROI Pool. Tương tự như R-CNN thì Fast R-CNN vẫn dùng selective search để lấy ra các region proposal. Tuy nhiên là nó không tách 2000 region proposal ra khỏi ảnh và thực hiện bài toán image classification cho mỗi ảnh. Fast R-CNN cho cả bức ảnh vào ConvNet (một vài convolutional layer + max pooling layer) để tạo ra convolutional feature map. Sau đó các vùng region proposal được lấy ra tương ứng từ convolutional feature map. Tiếp đó được Flatten và thêm 2 Fully connected layer (FCs) để dự đoán lớp của region proposal và giá trị offset values của bounding box.



Hình 2.14: Hình ảnh minh họa thuật toán Fast R - CNN[2]

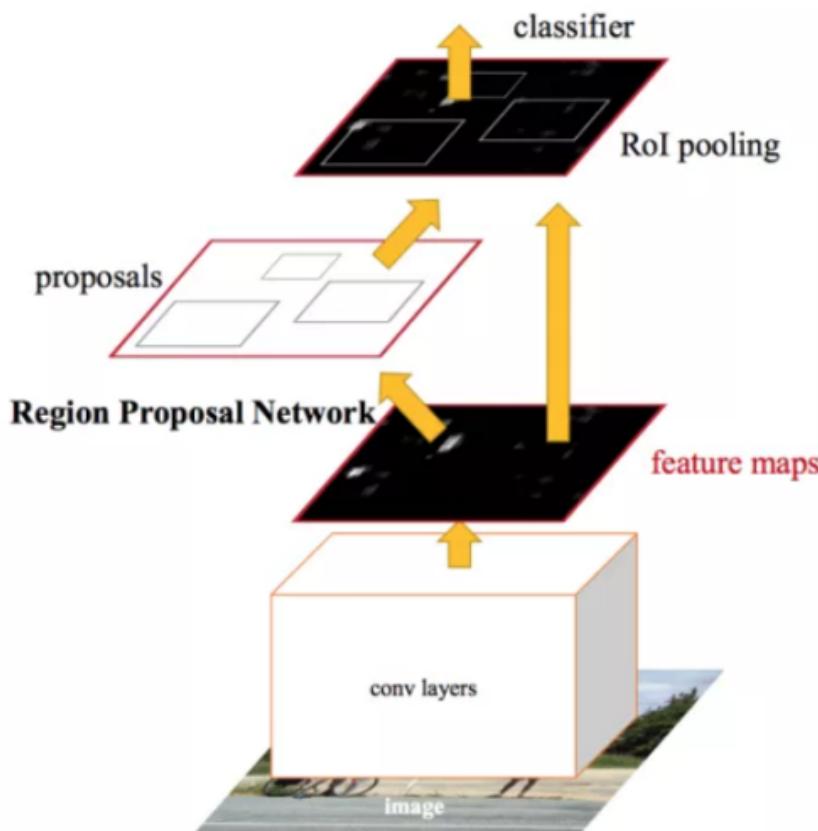
Fast R-CNN khác với R-CNN là nó thực hiện feature map với cả ảnh sau đó với lấy các region proposal ra từ feature map, còn R-CNN thực hiện tách các region proposal ra rồi mới thực hiện CNN trên từng region proposal. Do đó Fast R-CNN nhanh hơn đáng kể nhờ tối ưu việc tính toán bằng Vectorization (biểu diễn bài toán dưới dạng vector).

Tuy nhiên nhìn hình trên ở phần test time với mục Fast R-CNN thì thời gian tính region proposal rất lâu và làm chậm thuật toán.

=> Cần thay thế thuật toán selective search. Giờ người ta nghĩ đến việc dùng deep learning để tạo ra region proposal → Faster R-CNN ra đời.

2.2.3 Mô hình Faster R-CNN

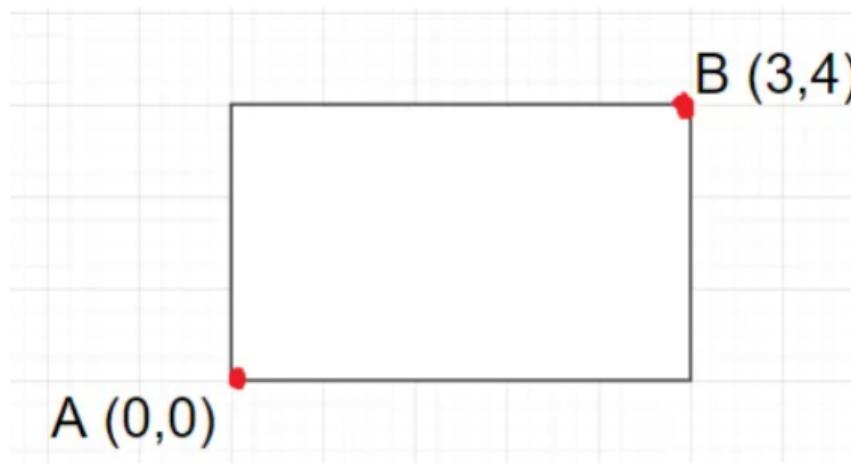
Faster R-CNN được giới thiệu vào đầu năm 2017. Đây là một bước tiến xa hơn khi thực hiện bước đột xuất vùng cách sử dụng ConvNet có tên là Region Proposal Network (RPN). Faster R-CNN không dùng thuật toán selective search để lấy ra các region proposal, mà nó thêm một mạng CNN mới gọi là RPN để tìm các region proposal.



Hình 2.15: Kiến trúc của mạng Faster R-CNN[2]

Đầu tiên cả bức ảnh được cho qua pre-trained model để lấy feature map. Sau đó feature map được dùng cho Region Proposal Network để lấy được các region proposal. Sau khi lấy được vị trí các region proposal thì thực hiện tương tự Fast R-CNN.

Region Proposal Network (RPN): Input của RPN là feature map và output là các region proposal. Ta thấy các region proposal là một hình chữ nhật.



Hình 2.16: Cách lấy toạ độ một bức ảnh

Mà một hình chữ nhật được xác định bằng 2 điểm ở 2 góc, ví dụ A(x_{min}, y_{min}) và B(x_{max}, y_{max}). Nhận xét: Khi RPN dự đoán ta phải rằng buộc $x_{min} < x_{max}$ và $y_{min} < y_{max}$. Hơn nữa các giá trị x,y khi dự đoán có thể ra ngoài khỏi bức ảnh => Cần một kĩ thuật mới để biểu diễn region proposal → Anchor ra đời. Ý tưởng là thay vì dự đoán 2 góc ta sẽ dự đoán điểm trung tâm (x_{center}, y_{center}) và width, height của hình chữ nhật. Như vậy mỗi anchor được xác định bằng 4 tham số (x_{center}, y_{center} , width, height). Vì không sử dụng Selective search nên RPN ban đầu cần xác định các anchor box có thể là region proposal, sau đó qua RPN thì chỉ output những anchor box chắc chắn chứa đối tượng.



Hình 2.17: Xác định tâm của bức ảnh bằng RPN [2]

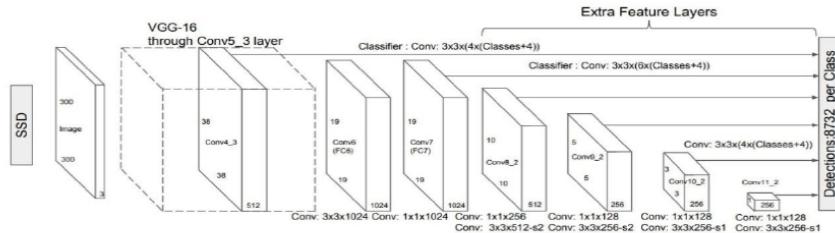
Ảnh bên trái kích thước $400 * 600$ pixel, tác động của anchor box màu xanh, cách nhau 16 pixel → có khoảng $(400*600)/(16*16) = 938$ tâm. Do các đối tượng trong ảnh có thể có kích thước và tỉ lệ khác nhau nên với mỗi tâm ta định nghĩa 9 anchors với kích thước $64x64$, $128x128$, $256x256$, mỗi kích thước có 3 tỉ lệ tương ứng: 1 : 1, 1 : 2 và 2 : 1. Giống như hình bên phải với tâm ở giữa 3 kích thước ứng với màu da cam, xanh lam, xanh lục và với mỗi kích thước có 3 tỉ lệ. → Số lượng anchor box giờ là $938 * 9 = 8442$ anchors. Tuy nhiên sau RPN ta chỉ giữ lại khoảng 1000 anchors box để thực hiện như trong Fast R-CNN.

Việc của RPN là lấy ra các region proposal giống như selective search chứ không phải là phân loại ảnh. Mô hình RPN khá đơn giản, feature map được cho qua Conv layer $3*3$, 512 kernels. Sau đó với mỗi anchor lấy được ở trên, RPN thực hiện 2 bước:

- Dự đoán xem anchor đây là foreground (chứa đối tượng) hay hình ảnh nền (không chứa đối tượng)
- Dự đoán 4 offset value cho x_{center}, y_{center} , width, height cho các anchor

Sau cùng dựa vào phần trăm dự đoán background RPN sẽ lấy N anchor (N có thể 2000, 1000, thậm chí 100 vẫn chạy tốt) để làm region proposal.

2.2.4 Mô hình SSD(Single Shot Detector)



Hình 2.18: Mô hình mạng SSD [2]

SSD đạt được sự cân bằng tốt giữa tốc độ và độ chính xác. SSD chỉ chạy một mạng CNN trên hình ảnh đầu vào một lần và tính toán một bản đồ các đặc trưng. Nay giờ chúng ta chạy một filter có kích thước nhỏ 3×3 trên toàn bộ bản đồ đặc trưng này để dự đoán các hộp giới hạn và xác suất phân loại. SSD cũng sử dụng các hộp neo ở các tỷ lệ khung hình khác nhau tương tự như Faster R-CNN. Để xử lý tỷ lệ, SSD dự đoán các hộp giới hạn sau nhiều lần co dãn.

2.2.5 Kết luận chương

Qua phần giới thiệu và đánh giá ở trên chúng ta có rất nhiều phương pháp để phát hiện phương tiện giao thông. Nhưng do tốc độ di chuyển của các phương tiện giao thông thường cao nên cần một phương pháp có tốc độ xử lý nhanh và độ chính xác tốt trong điều kiện ánh sáng tốt.

CHƯƠNG 3. PHƯƠNG PHÁP ĐỀ XUẤT

3.1 Bài toán nhận diện biển báo và đèn tín hiệu giao thông

3.1.1 Tổng quan bài toán

Nhận diện biển báo và đèn tín hiệu giao thông là một bài toán khó và phức tạp khi cần xác định vị trí và phân loại các loại biển báo và đèn tín hiệu giao thông. Hệ thống này bao gồm ba bước xử lý chính:

1. Thu nhận ảnh từ các hệ thống camera giao thông và thực hiện tiền xử lý dữ liệu đầu vào.
2. Sử dụng một mô hình phát hiện đã được huấn luyện để phát hiện (YOLO, R-CNN, SSD...) và trả về kết quả bao gồm hộp giới hạn và đối tượng xuất hiện trong các hộp giới hạn đó.
3. Thực hiện hậu xử lý để loại bỏ các hộp chồng chéo, theo dõi các loại biển báo, đèn tín hiệu...

3.1.2 Các điều kiện ràng buộc

Để thu được kết quả phát hiện chính xác cao các hệ thống camera cần được đặt tại các vị trí phù hợp như giữa trực đường chính để có góc nhìn rộng tránh các trường hợp các biển báo hay đèn tín hiệu bị che khuất, các vị trí có ánh sáng tốt.

Các hệ thống xử lý trung tâm cần được hỗ trợ các máy tính xử lý có nhiều CPU hoặc GPU giúp cho việc phát hiện các biển báo, đèn tín hiệu nhanh và chính xác đảm bảo thời gian thực.

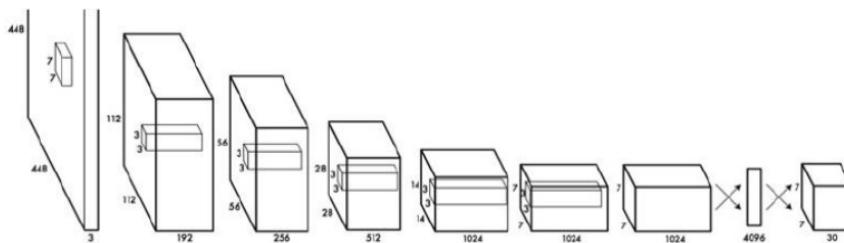
3.2 Hệ thống phát hiện đối tượng thời gian thực YOLO

3.2.1 Tổng quan hệ thống phát hiện đối tượng YOLO.

YOLO là một phương pháp phát hiện đối tượng thời gian thực hiện đại nhất hiện nay. Với GPU Titan X mô hình YOLO có tốc độ 30 FPS với độ chính xác trung bình 57.9% trên tập dữ liệu tiêu chuẩn COCO. YOLO rất phù hợp cho các bài toán đòi hỏi tốc độ xử lý theo thời gian thực như giám sát giao thông, xe tự lái, các hệ thống giám sát an ninh.

3.2.2 Kiến trúc của mô hình YOLO qua các phiên bản

a, YOLO version 1



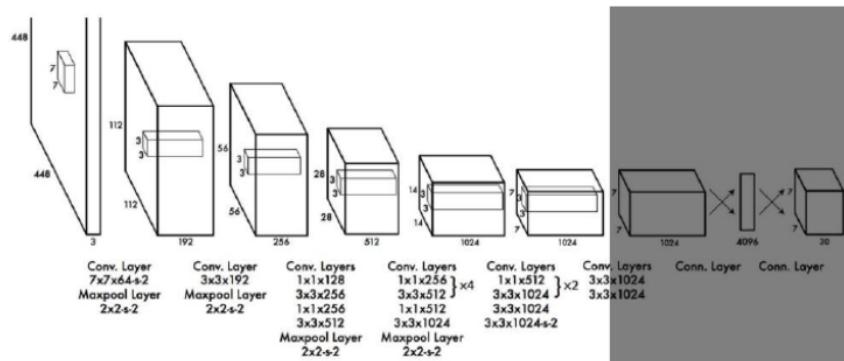
Hình 3.1: YOLO version 1 [3]

YOLOv1 có 24 lớp convolutional sau là 2 lớp fully connected (FC). Một vài lớp convolutional sử dụng các lớp giảm có kích thước 1×1 để giảm độ sâu của các features map. Đối với lớp convolutional cuối cùng, nó xuất ra một tensor có kích thước ($7 \times 7 \times 1024$). Tensor sau đó được dàn phẳng ra. Sử dụng 2 fully connected dưới dạng hồi quy tuyến tính tạo ra output có kích thước $7 \times 7 \times 30$ tương ứng với dự đoán 2 hộp giới hạn cho mỗi ô và 20 lớp dự đoán như ví dụ trên.

b, YOLO version 2

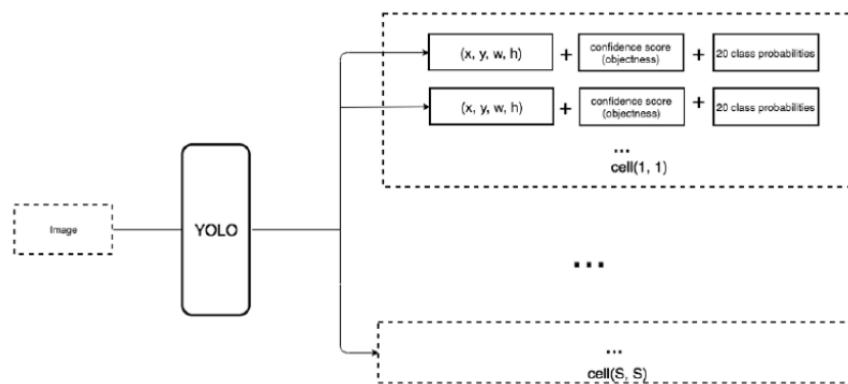
Để cải thiện độ chính xác YOLOv2 đã thực hiện cải tiến các phương pháp sau

- Loại bỏ các lớp fully connected chịu trách nhiệm dự đoán hộp giới hạn



Hình 3.2: Loại bỏ phần kết nối đầy đủ trên YOLOv2 [3]

- YOLO di chuyển lớp dự đoán từ cấp độ ô lên cấp độ hộp giới hạn. Bây giờ mỗi dự đoán bao gồm 4 tham số cho hộp giới hạn, 1 điểm tin cậy của hộp và C xác suất lớp, tức là với 5 hộp giới hạn với 25 tham số cho mỗi hộp giới hạn ta có 125 tham số cho mỗi ô.

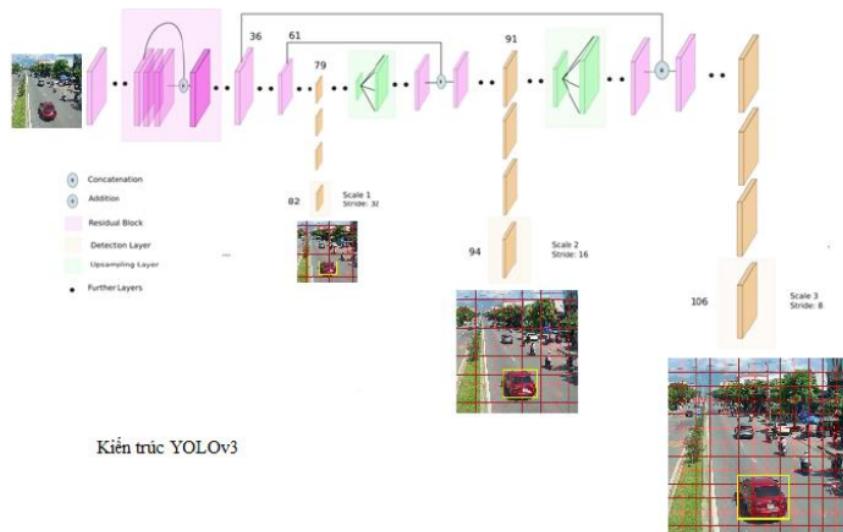


Hình 3.3: Kết quả dự đoán của YOLO[3]

- Để tạo ra các dự đoán với kích thước $7 \times 7 \times 125$, chúng ta sẽ thay thế lớp convolutional cuối cùng bằng ba filter có kích thước 3×3 cho mỗi đầu ra có 1024 giá trị. Sau đó lớp convolutional cuối cùng ta áp dụng filter có kích thước 1×1 để chuyển từ $7 \times 7 \times 1024$ output về $7 \times 7 \times 125$.
- Thay đổi kích thước ảnh đầu vào từ 448×448 xuống 416×416
- Loại bỏ một lớp pooling để làm cho đầu ra của mạng có kích thước 13×13 (thay vì 7×7)

c, YOLO version 3

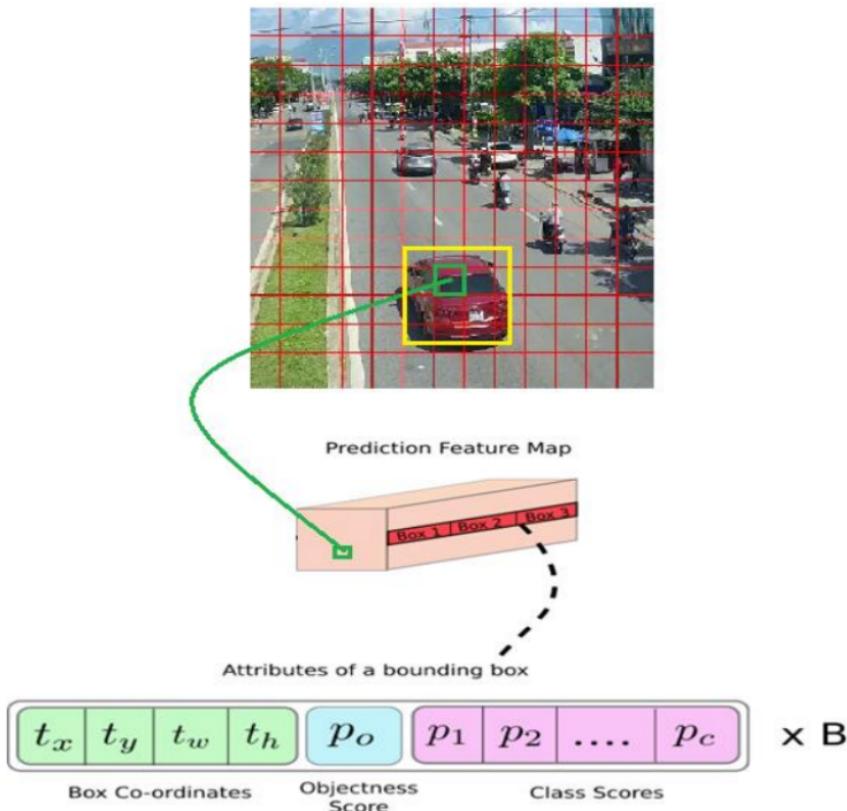
YOLOv3 sử dụng một biến thể của Darknet[25, 26], ban đầu có 53 lớp được đào tạo trên Imagenet. Đối với nhiệm vụ phát hiện, 53 lớp khác được xếp chồng lên nó, tạo cho chúng ta một kiến trúc cơ bản hoàn toàn dựa trên 106 lớp cho YOLOv3



Hình 3.4: Mô hình YOLO version 3

Hình dạng kernel của phát hiện là $1 \times 1 \times (B \times (5+C))$. Ở đây B là số lượng hộp giới hạn một ô trên ma trận đặc trưng có thể dự đoán, 5 là thuộc tính của hộp giới hạn

và điểm tin cậy và C là số lớp. Ma trận đặc trưng được tạo ra bởi kernel này có chiều rộng và chiều cao giống hệt nhau của ma trận đặc trưng trước đó và có các thông tin đọc theo chiều sâu như mô tả hình bên dưới



Hình 3.5: Kết quả dự đoán của YOLOv3

d, YOLO version 4

YOLOv4 áp dụng ý tưởng của CSPBlock, thay thế Residual Block thông thường của YOLOv3 thành CSPResBlock, đồng thời đổi activation function từ LeakyReLU thành Mish, tạo nên CSPDarkNet53.

Trong các bài Classification, ở layer cuối ta thường hay sử dụng DropOut để làm giảm hiện tượng Overfitting. Nhưng trong Convolution thì việc bỏ đi random một số vị trí ở trong feature map có vẻ không hợp lý lắm. Vì các vị trí ở cạnh nhau trong feature map có tương quan cao với nhau, nên việc bỏ đi các vị trí một cách random dường như sẽ không đem lại nhiều hiệu quả. DropBlock sẽ bỏ đi nhóm vị trí trong feature map thay vì chỉ bỏ đi một vị trí

Cũng giống YOLOv3, YOLOv4 có thêm neck để thực hiện phát hiện vật thể ở trên những scale khác nhau. YOLOv4 sử dụng 2 thành phần cho neck: SPP và PAN.

YOLOv4 áp dụng Label Smoothing trong việc classification cho vật thể. Ý tưởng về Label Smoothing lần đầu được giới thiệu trong Inception-v2. Việc này làm giảm

sự tự tin thái quá của model khi dự đoán một class, từ đó giảm nhẹ overfitting của model.

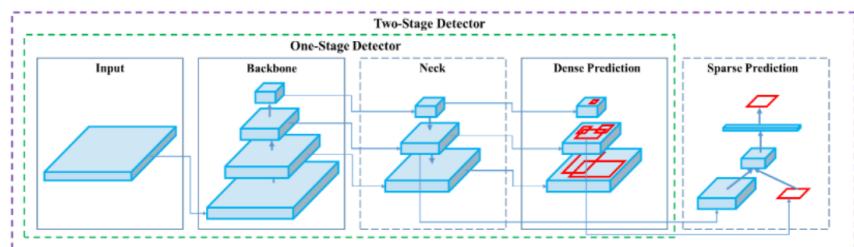
e, YOLO version 5

YOLOv5 là một phần mở rộng tự nhiên của YOLOv3 PyTorch bởi 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. Những cải tiến này ban đầu được gọi là YOLOv4 nhưng do việc phát hành gần đây của YOLOv4 trong khuôn khổ Darknet, để tránh xung đột phiên bản, nó đã được đổi tên thành YOLOv5.

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

f, Cấu trúc nhận diện vật thể của YOLOv5

Cấu trúc nhận diện vật thể của YOLOv5 thường có 3 phần được thể hiện ở hình bên dưới.



Hình 3.6: Cấu trúc nhận diện vật thể của YOLOv5[3]

- Backbone: Backbone là 1 mô hình pre-train của 1 mô hình học chuyển (transfer learning) khác để học các đặc trưng và vị trí của vật thể. Các mô hình học chuyển thường là VGG16, ResNet-50,...
- Head: Phần head được sử dụng để tăng khả năng phân biệt đặc trưng để dự đoán class và bounding-box. Ở phần head có thể áp dụng 1 tầng hoặc 2 tầng:
 - Tầng 1: Dense Prediction, dự đoán trên toàn bộ hình với các mô hình RPN, YOLO, SSD,...
 - Tầng 2: Sparse Prediction dự đoán với từng mảng được dự đoán có vật thể với các mô hình R-CNN series,..
- Neck: Ở phần giữa Backbone và Head, thường có thêm một phần Neck. Neck thường được dùng để làm giàu thông tin bằng cách kết hợp thông tin giữa quá trình bottom-up và quá trình top-down (do có một số thông tin quá nhỏ khi đi

qua quá trình bottom-up bị mất mát nên quá trình top-down không tái tạo lại được).

g, Đặc điểm của YOLOv5 với các phiên bản trước của YOLO

Vì YOLOv5 được triển khai trong PyTorch ban đầu nên nó được hưởng lợi từ hệ sinh thái PyTorch đã được thiết lập: hỗ trợ đơn giản hơn và triển khai dễ dàng hơn. Hơn nữa, là một khung nghiên cứu được biết đến rộng rãi hơn, việc lặp lại trên YOLOv5 có thể dễ dàng hơn cho cộng đồng nghiên cứu rộng lớn hơn. Điều này cũng làm cho việc triển khai đến các thiết bị di động đơn giản hơn vì mô hình có thể được biên dịch sang ONNX và CoreML một cách dễ dàng.

Khả năng đào tạo cũng như khả năng suy luận rất là nhanh, độ chính xác cao. Cuối cùng YOLOv5 có dung lượng nhỏ. Cụ thể, một tệp trọng số cho YOLOv5 là 27 megabyte. Tệp trọng số cho YOLOv4 (với kiến trúc Darknet) là 244 megabyte. YOLOv5 nhỏ hơn gần 90% so với YOLOv4. Điều này có nghĩa là YOLOv5 có thể được triển khai cho các thiết bị nhúng dễ dàng hơn nhiều.

CHƯƠNG 4. CÀI ĐẶT CHƯƠNG TRÌNH

4.1 Chuẩn bị bộ Dataset

Bộ dữ liệu được sử dụng trong bài toán phát hiện biển báo và đèn tín hiệu giao thông sử dụng mô hình thuật toán YOLO bao gồm 7500 hình ảnh về đèn tín hiệu và biển báo giao thông của nhiều bộ dữ liệu khác nhau được thu thập trong nhiều điều kiện khác nhau. Traffic Light dataset là bộ dữ liệu đèn tín hiệu giao thông của Trung Quốc, bao gồm 3000 ảnh được chụp ở các điều kiện khác nhau và các cung đường khác nhau. Bộ dữ liệu biển báo giao thông Việt Nam thu thập từ các camera hành trình chứa 4500 hình ảnh và được phân loại theo 7 loại chính: cấm ngược chiều, cấm dừng và đỗ, cấm rẽ, giới hạn tốc độ, nguy hiểm, hiệu lệnh và một vài loại biển báo còn lại.



Hình 4.1: Bộ dữ liệu thu thập được

Với lượng dữ liệu hình ảnh trên, đây là một bộ dữ liệu không hề ít, ở lần thử đầu tiên train thuật toán YOLOv5 với bộ dữ liệu này, mô hình cho kết quả như sau:

- Ở điều kiện thời tiết tốt, trời sáng, các biển báo hay đèn tín hiệu nằm ở vị trí thuận lợi, mô hình đem lại kết quả với độ chính xác khá cao, nhận diện được hầu hết các đối tượng cần thiết trong ảnh.



Hình 4.2: Kết quả thu được ở điều kiện tốt

- Ở các điều kiện thời tiết xấu (trời mưa, nắng quá to,...), ánh sáng môi trường kém(quá tối hoặc quá sáng), các biển báo bị mờ hoặc chụp ở góc nghiêng,... thì mô hình đem lại kết quả không quá cao, thậm chí đa số sẽ không nhận diện được hoặc nhận diện sai các đối tượng biển báo hay đèn tín hiệu.



Hình 4.3: Kết quả thu được ở điều kiện sương mù

Dễ dàng nhận ra vấn đề khi thực hiện train thuật toán YOLOv5 với tập dữ liệu nguyên gốc ban đầu: Kết quả huấn luyện có độ chính xác rất cao trên tập huấn luyện và tập kiểm tra. Nhưng khi áp dụng vào thực tế với điều kiện sương mù, bụi, ánh sáng yếu,... thì độ chính xác giảm đi rất nhiều. Qua quá trình tìm hiểu, em nhận ra những nguyên nhân chính dẫn đến vấn đề kết quả không như mong đợi này đó là:

- Đặc thù bài toán hỗ trợ con người trong quá trình lái xe nói chung và bài toán nhận diện biển báo, đèn tín hiệu giao thông nói riêng, yêu cầu thực hiện trong mọi loại môi trường thời tiết, mọi loại địa hình, mọi thời điểm trong ngày. Đặc

biệt càng trong điều kiện khó khăn trong quan sát thì càng yêu cầu hệ thống phải hỗ trợ ở mức tốt nhất cho con người. Ví dụ một người lái xe dưới trời mưa, lượng mưa ngoài trời sẽ ảnh hưởng khá lớn đến tầm nhìn các biển báo giao thông; khi lái xe dưới thời tiết quá nắng, ánh nắng đôi khi gây chói mắt và ảnh hưởng đến việc quan sát biển báo; hoặc lái xe vào buổi đêm, tài xế dễ bị buồn ngủ dẫn đến không để ý các biển báo, đèn tín hiệu trên đường,...Những lúc này hệ thống hỗ trợ đưa ra những cảnh báo, nhắc nhở sẽ là một tính năng vô cùng hữu ích. Vì vậy để hệ thống hỗ trợ con người đem lại sự khả dụng cao nhất, thì yêu cầu hệ thống này phải chạy được ở trong nhiều loại điều kiện, nhất là những điều kiện khó khăn. Nếu chỉ chạy được ở trong những điều kiện tốt, hoặc chỉ trong điều kiện xấu, thì tính khả dụng xem như bằng không.

- Bộ dữ liệu ảnh ở trên tuy lớn nhưng lại gấp một số vấn đề sau:
 - Đa số ảnh chụp vào ban ngày, lượng ánh sáng tốt, dễ dàng trong việc quan sát các đối tượng.
 - Điều kiện thời tiết chủ yếu là nắng và trời râm, khi train với tập ảnh này sẽ luyện cho mô hình khả năng nhận diện biển báo, đèn tín hiệu ở những hoàn cảnh khắc nghiệt như ngược hướng nắng, trời sương mù, nhưng cũng có nhược điểm là mô hình chỉ hoạt động tốt ở hai điều kiện trên, điều này là chưa đủ vì quá trình tham gia giao thông, đặc biệt ở Việt Nam, đất nước nằm trong vùng khí hậu nhiệt đới và á nhiệt đới có gió mùa, các điều kiện thời tiết hết sức đa dạng, thì mô hình trên cần phải chạy được ở nhiều loại điều kiện thời tiết hơn nữa.
 - Phân phối của tập huấn luyện khác so với thực tế, đó là ở tập huấn luyện, ảnh chứa biển báo và ảnh chứa đèn tín hiệu chiếm tỉ lệ 70:30, tuy nhiên khi sử dụng vào thực tế, lượng ảnh chứa đèn tín hiệu lại khá lớn và chiếm tỉ lệ 50:50

Và ngoài ra vẫn tồn tại một số nguyên nhân phụ khác dẫn tới thuật toán hoạt động không được như kì vọng. Tuy nhiên, dựa vào các nguyên nhân chính mà đã nêu ở trên, em đã tìm hiểu và áp dụng một số phương án để khắc phục các vấn đề, đồng thời cải thiện khả năng chính xác của mô hình qua việc tiền xử lý dữ liệu đầu vào. Phần sau đây em sẽ trình bày chi tiết về các phương án đã sử dụng cho việc tiền xử lý bộ dữ liệu của mình.

4.1.1 Thay đổi độ tương phản của ảnh

Trong những trường hợp dữ liệu là một hình ảnh chụp trong điều kiện thời tiết đẹp, ánh sáng tốt, em đã dùng phương pháp thay đổi độ tương phản của ảnh để làm

giàu thêm bộ dữ liệu của mình như sau:

- Tăng độ tương phản để tạo ra một môi trường với lượng ánh sáng lớn, để áp dụng vào trong các trường hợp khi tham gia giao thông, ánh nắng mặt trời quá mạnh gây khó khăn trong việc nhận diện, đặc biệt là biển báo hay đèn tín hiệu có màu trắng, vàng.
- Giảm độ tương phản: tương tự như cách trên, từ ảnh gốc ta sẽ giảm độ tương phản để tạo thêm một ảnh mới. Khi luyện tập với những ảnh này, thuật toán sẽ tăng khả năng nhận diện vật thể ở những điều kiện ánh sáng kém như buổi chiều tối hay buổi đêm.

Ví dụ, từ một ảnh gốc ban đầu trong bộ dữ liệu ban đầu, ảnh được chụp lúc ban ngày, thời điểm ánh sáng tốt, biển báo và đèn tín hiệu trong ảnh rất rõ ràng:



Hình 4.4: Một ảnh trong bộ dữ liệu được chụp dưới điều kiện ánh sáng tốt

Ta sẽ sử dụng hàm `adjust_image_gamma` để xử lý vấn đề trên, với tham số truyền vào là giá trị gamma để điều chỉnh độ tương phản

```
def adjust_image_gamma(image, gamma = 1.0):
    image = np.power(image, gamma)
    max_val = np.max(image.ravel())
    image = image / max_val * 255
    image = image.astype(np.uint8)
    return image
```

Ví dụ khi cần tăng độ sáng để giả lập điều kiện ánh nắng từ mặt trời gây chói, ta chỉnh giá trị gamma thấp (ở ví dụ này giá trị gamma = 0.5), ngược lại, để giảm độ sáng xuống giả lập môi trường trời chiều tối, sử dụng một giá trị gamma lớn (ví dụ bên dưới gamma = 6), ta thu được kết quả như 2 ảnh phía dưới:

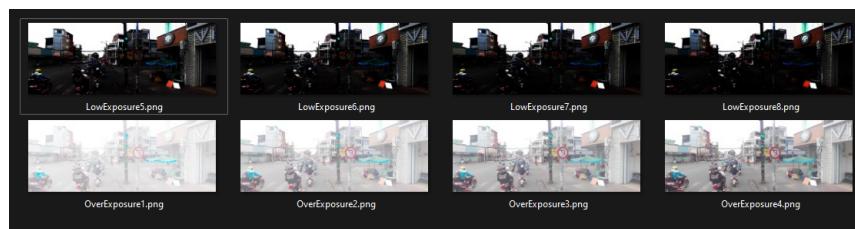


Hình 4.5: Bức ảnh sau khi tăng độ tương phản



Hình 4.6: Bức ảnh sau khi giảm độ tương phản

Thay đổi các giá trị gamma, ta thu được một tập ảnh với các điều kiện sáng khác nhau từ một ảnh gốc ban đầu:



Hình 4.7: Tập ảnh với các điều kiện sáng khác nhau

4.1.2 Làm mờ ảnh

Phép làm mờ ảnh đem lại một số lợi ích như:

- Giảm nhiễu trong ảnh
- Làm trơn ảnh (smooth). Việc làm trơn ảnh sẽ giảm sắc nét của cạnh, thay vào đó, vùng trơn sẽ lan ra

Cụ thể trong bài toán nhận diện biển báo và đèn tín hiệu giao thông, từ bức ảnh ở tập huấn luyện ban đầu, ta có thể sử dụng phương pháp làm mờ để tạo ra một bức ảnh mới. Việc này giúp model sau này sẽ nhận ra được các đối tượng cần thiết kể cả trong môi trường khó quan sát như trời mưa hay có sương mù.

Ví dụ từ ảnh gốc ban đầu, ta sử dụng hàm `blur()` của thư viện OpenCV, hàm này làm mờ ảnh bằng cách thiết lập giá trị trên bộ lọc bằng $1/(W \cdot H)$, tức là nếu Width và Height của bộ lọc là 5, thì giá trị trên cửa sổ convolve là $1/25$.



Hình 4.8: Ảnh gốc khi chưa chạy qua bộ lọc làm mờ



Hình 4.9: Ảnh gốc sau khi chạy qua bộ lọc làm mờ

Tương tự như cách xử lý ở phần thay đổi độ tương phản, ta sẽ thực hiện với một tập (W, H) chạy từ 5-9 để tạo ra một bộ ảnh với các độ mờ khác nhau từ một ảnh gốc ban đầu:



Hình 4.10: Bộ ảnh với các độ blur khác nhau tạo ra từ ảnh gốc

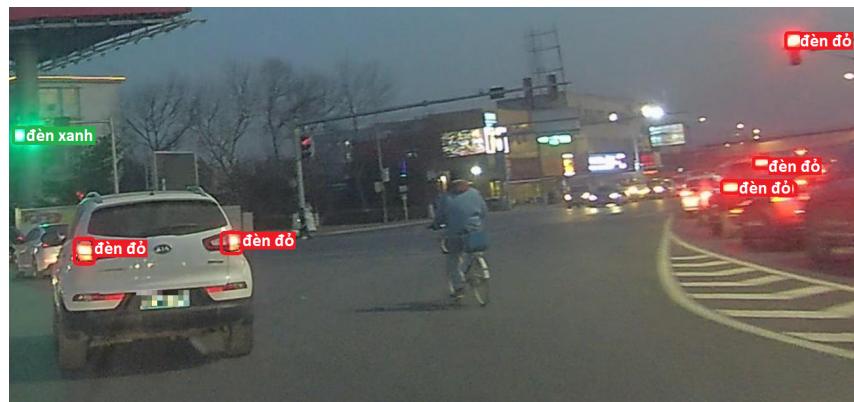
4.1.3 Loại bỏ phần dữ liệu không đủ điều kiện

Sau khi thực hiện một số công việc để bổ sung các điều kiện khác nhau vào bộ dữ liệu, ta thu được một bộ dữ liệu mới với kích thước lớp gấp chục lần bộ dữ liệu ban đầu. Tuy nhiên trong đó có vô số ảnh không đảm bảo điều kiện như sau khi chạy qua bước điều chỉnh độ tương phản, sẽ có những ảnh quá sáng hoặc quá tối,

hay khi chạy qua bộ làm mờ, có những ảnh bị làm mờ đi nhiều quá, hay cũng có những kết quả 2 bức ảnh không có sự khác biệt quá lớn sau khi chạy qua những bộ lọc trên. Những trường hợp này sẽ gây chậm quá trình huấn luyện và tác dụng làm tăng độ chính xác của thuật toán gần như bằng 0, ta sẽ phải thực hiện xoá bỏ đi những dữ liệu dư thừa này. Sau khi đã xoá bỏ được những hình ảnh không đủ điều kiện ở trên, ta sẽ thu được bộ ảnh với các điều kiện ánh sáng và độ mờ khác nhau, điều này giúp cho thuật toán sau này khả thi trong các điều kiện sáng, tối hay thời tiết xấu như mưa, sương mù,....

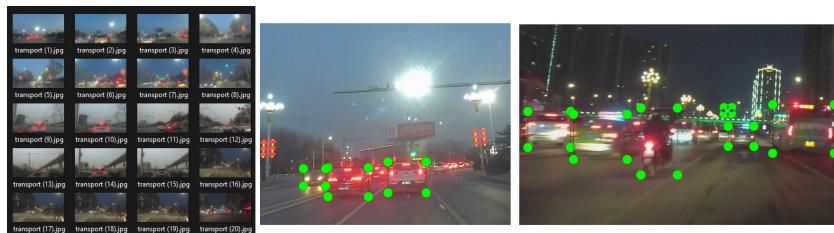
4.1.4 Bổ sung bộ dữ liệu về các phương tiện giao thông để phục vụ cho việc hậu xử lý dữ liệu

Việc nhận diện đèn tín hiệu ở điều kiện ban đêm là hết sức quan trọng và cần thiết, tuy nhiên nó cũng đem lại rất nhiều khó khăn vì vào ban đêm, môi trường bên ngoài có rất nhiều nguồn sáng khác nhau. Bên cạnh những đối tượng là đèn giao thông mà mô hình của chúng ta cần nhận diện, còn có những nguồn sáng khác như đèn đường, đèn từ cửa hàng, nhà dân, biển quảng cáo,... và nhiều nhất là đèn từ các phương tiện giao thông, phổ biến nhất là xe tải, ô tô, xe máy,... Những nguồn sáng này gần như không có sự khác biệt với các đèn tín hiệu giao thông, vì vậy nếu không có cách xử lý với những nguồn sáng từ các phương tiện này thì tỉ lệ nhận diện các đèn tín hiệu sẽ rất lớn.



Hình 4.11: Dễ dàng nhầm lẫn đèn từ các phương tiện giao thông thành đèn tín hiệu giao thông

Để giải quyết vấn đề trên, em đã tận dụng khả năng nhận diện nhiều đối tượng một lúc của model YOLOv5 bằng cách bổ sung một bộ dữ liệu về các phương tiện giao thông, cụ thể là 3 loại phương tiện phổ biến : xe tải, ô tô, xe máy. Việc nhận diện những phương tiện này sẽ góp phần cho việc xử lý ảnh đầu ra mà em sẽ nêu ở phần sau.



Hình 4.12: Bộ dữ liệu bổ sung cho việc nhận diện phương tiện giao thông, phục vụ hậu xử lý sau này

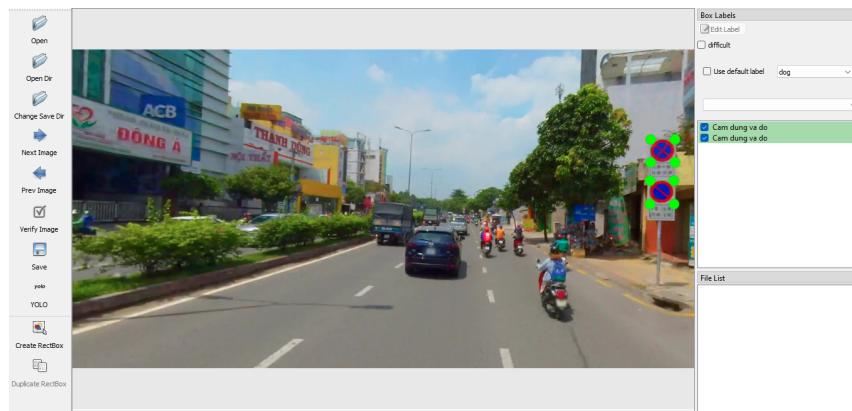
4.1.5 Cấu trúc bộ dữ liệu phục vụ cho quá trình huấn luyện

Trong folder train_data sẽ có 2 cấu trúc thư mục là images và labels, thư mục images để chứa các hình ảnh biển báo giao thông và thư mục labels dùng để xác định được các nhãn tên gắn trên hình ảnh để hỗ trợ chuẩn đoán hay xác định chính xác dấu hiệu



Hình 4.13: Cấu trúc thư mục train_data

Chúng ta sử dụng tool LabelImg để đánh nhãn vật thể trong hình. Đây là một công cụ hỗ trợ mạnh mẽ trong việc đánh nhãn vật thể để train các model detection như YOLO.



Hình 4.14: Gán nhãn hình ảnh bằng LabelImg

File gán nhãn .txt có format như sau:

- Mỗi hàng sẽ là một đối tượng.
- Mỗi hàng sẽ có format: class x_center y_center width height.
- Toạ độ của các box sẽ được normalized (từ 0-1) theo format xywh
- Class sẽ bắt đầu từ 0.

4.2 Thiết lập môi trường huấn luyện nhận dạng biển báo và đèn tín hiệu giao thông

Yêu cầu Pytorch version 1.5 trở lên, Python version 3.7 và CUDA

Để bắt đầu với YOLOv5, trước tiên sao chép kho lưu trữ YOLOv5 và cài đặt các tính năng phụ thuộc.

```
[ ] %cd /content/drive/MyDrive
!git clone https://github.com/ultralytics/yolov5 # clone

[ ] %cd /content/drive/MyDrive/yolov5
%pip install -qr requirements.txt # install

[ ] import torch
from drive.MyDrive.yolov5 import utils
display = utils.notebook_init() # checks
```

Hình 4.15: Clone model và cài đặt các requirements

Tạo file /yolov5/data/custom_data.yaml để chỉ định vị trí của thư mục hình ảnh YOLOv5, thư mục nhãn YOLOv5 và thông tin về các lớp custom.

```

train: ./yolov5/data/train_data/images/train/ # train images (relative to 'path') 128 images
val: ./yolov5/data/train_data/images/val/ # val images (relative to 'path') 128 images

# Classes
nc: 10 # number of classes
names: [
'den do',
'den xanh',
'den vang',
'cam nguoc chieu',
'cam dung va do',
'cam re',
'gioi han toc do',
'cam con lai',
'nguy hiem',
'hiem lenh'
] # class names

```

Hình 4.16: Cấu hình lại file yaml

4.3 Traning

Để bắt đầu huấn luyện, ta chạy the training command theo tùy chọn sau:

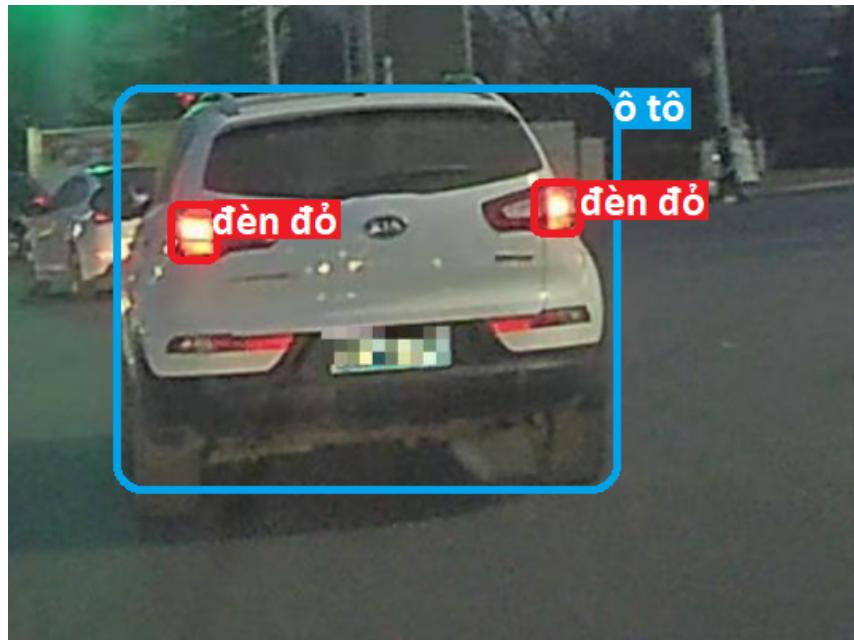
- img: xác định kích thước hình ảnh đầu vào
- batch: số ảnh để load vào (16-32) một lần
- epochs: số lần học
- data: đặt đường dẫn đến tệp yaml
- cfg: chỉ định cấu hình mô hình
- weights: chỉ định một đường dẫn tùy chỉnh đến weights.
- name: tên kết quả
- nosave: chỉ lưu điểm kiểm tra cuối cùng
- cache: hình ảnh trong bộ nhớ cache để train nhanh hơn

4.4 Xử lý các trường hợp nhận diện nhầm đèn tín hiệu với đèn của các phương tiện giao thông

Như đã đề cập ở phần trước, việc di chuyển trong điều kiện ban đêm, khi mà ngoài đèn tín hiệu còn rất nhiều đèn sáng đến từ nhiều nguồn khác nhau, phổ biến nhất là đèn từ các phương tiện giao thông như xe tải, ô tô, xe máy. Và model sau khi huấn luyện của chúng ta đang bị nhầm với những nguồn đèn này. Vì vậy để giải quyết vấn đề trên, trong bộ dataset ngoài dữ liệu về đèn tín hiệu và biển báo giao thông, em còn bổ sung thêm dữ liệu về các phương tiện giao thông để phục vụ cho phần hậu xử lý dữ liệu.

Dễ dàng nhận thấy trong trường hợp có ánh đèn phát ra từ phương tiện giao thông, model sẽ trả về kết quả là bounding box của hai đối tượng: phương tiện và đèn tương ứng. Và vùng nhận diện của đèn thường sẽ nằm 75-100% diện tích trong

vùng nhận diện của phương tiện:



Hình 4.17: Model nhận diện ô tô và đèn

Ta sẽ xây dựng một hàm để lọc các trường hợp này và loại bỏ khả năng đây là đèn tín hiệu giao thông.

Lấy tọa độ của vật thể dựa trên file best.pt đã huấn luyện

```

model = torch.hub.load(
    'ultralytics/yolov5',
    'custom',
    path='./best.pt')
detections = model(frame)

results = numpy.array(
    detections.pandas().xyxy[0].to_dict(orient="records"))
-----
```

pTransport = Polygon([
 (resultTransport.xmin, resultTransport.xmax),
 (resultTransport.xmax, resultTransport.ymin),
 (resultTransport.xmax, resultTransport.ymax),
 (resultTransport.xmin, resultTransport.ymax),])

pLight = Polygon([
 (resultLight.xmin, resultLight.xmax),

```
( resultLight .xmax , resultLight .ymin ) ,
( resultLight .xmax , resultLight .ymax ) ,
( resultLight .xmin , resultLight .ymax ), ] )
```

Gọi đến hàm checkTrafficLight, kiểm tra xem đối tượng đèn là đèn tín hiệu hay đèn phương tiện giao thông

```
isTrafficLight = checkTrafficLight(pTransport, pLight)

if isTrafficLight:
    #Xử lý: thông báo cho người dùng có đèn tín hiệu trên đường
else:
    #Loại bỏ trường hợp pLight là đèn tín hiệu

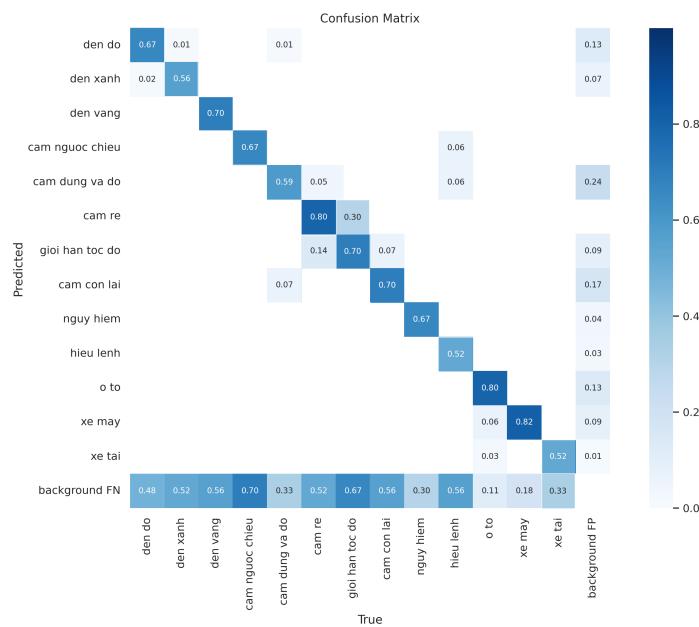
def checkTrafficLight(pTransport, pLight):
    intersection = pTransport.intersection(pLight);
    if intersection.area == 0.0:
        return True
    if (intersection.area)/(pLight.area) >= 0.75:
        return True
    else:
        return False
```

4.5 Kết quả chương trình

Sau quá trình train, trọng số (weights) của model Yolov5 sẽ được lưu trong thư mục run/train/exp/weights, đó là weights của epoch tốt nhất và best.pt và epoch cuối cùng last.pt.

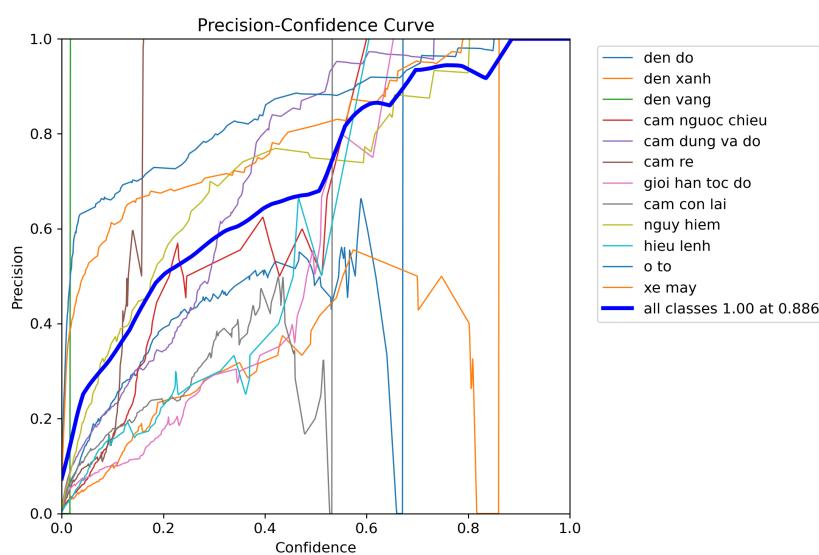
Kết quả của việc huấn luyện được mô tả dưới đây:

Confusion matrix (ma trận nhầm lẫn), ma trận này sẽ cho ta biết được tỉ lệ dự đoán chính xác hoặc tỉ lệ nhầm lẫn giữa các đối tượng với nhau. Ví dụ theo sơ đồ trên ta thấy biển cấm rẽ có tỉ lệ nhận diện chính xác là 0.56 hay 56%, tuy nhiên tỉ lệ nhận diện nhầm biển này với biển giới hạn tốc độ cũng đang cao là 30%, hay biến giới hạn tốc độ cũng đang gây nhiễu khi model nhận diện là cấm rẽ tỉ lệ nhầm lẫn 14%... Dựa vào kết quả trong ma trận này ta đưa ra được đánh giá tổng quát mô hình nhận diện các vật thể cần thiết ở tỉ lệ khá tốt.

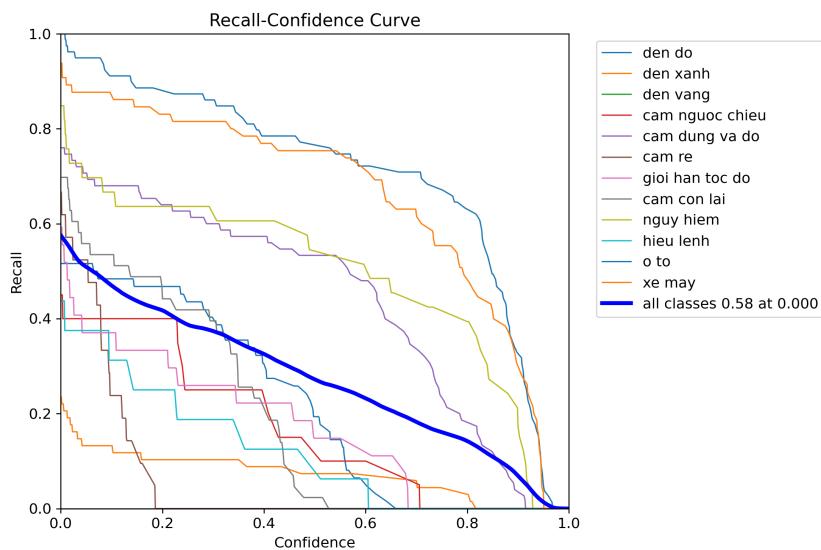


Hình 4.18: Confusion_matrix

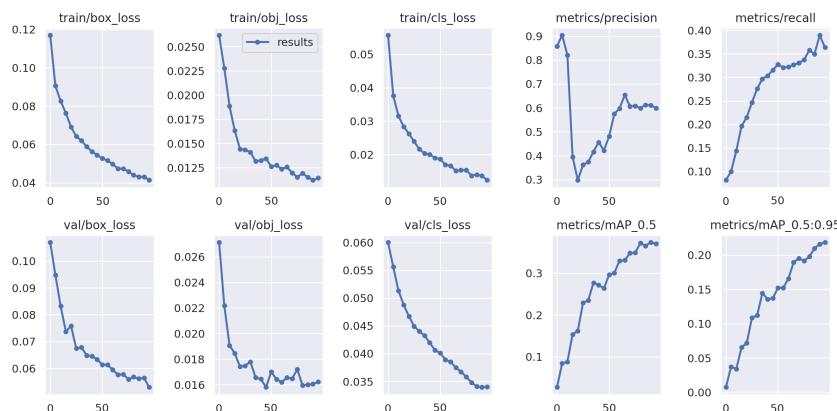
Sơ đồ sự phụ thuộc giữa Precision-Confidence và Recall-Confidence:



Hình 4.19: Sơ đồ P_curve

**Hình 4.20:** Sơ đồ R_curve

Dựa trên biểu đồ kết quả trên, ta đưa ra nhận xét kết quả model dự đoán khá tốt khi các giá trị box_loss, obj_loss, cls_loss đều giảm dần qua các lần train, còn các giá trị mAP tăng dần qua các lần train.

**Hình 4.21:** Kết quả huấn luyện

Thực hiện detect ta thu được một số kết quả sau:



canh bao: cam dung va do

Hình 4.22: Phát hiện biển báo Cấm dừng, đỗ



Hình 4.23: Phát hiện đèn xanh, biển cấm rẽ, biển cấm ngược chiều



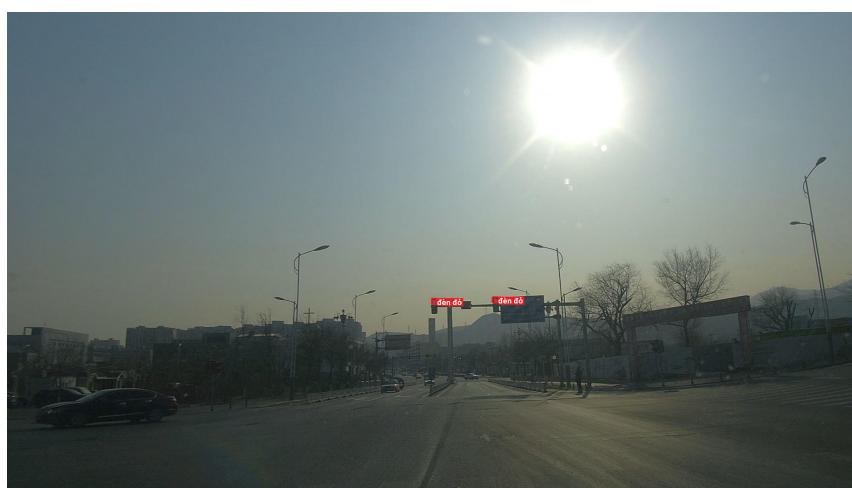
Hình 4.24: Phát hiện đèn đỏ và biển báo Cấm đi ngược chiều



Hình 4.25: Phát hiện biển báo Cấm dừng, đỗ và cấm đi ngược chiều trong điều kiện ban đêm



Hình 4.26: Phát hiện đèn tín hiệu trong điều kiện ban đêm



Hình 4.27: Phát hiện đèn tín hiệu trong điều kiện nắng gắt

CHƯƠNG 5. KẾT LUẬN

5.1 Kết luận

Sau một thời gian tìm hiểu đề tài, em đã phần nào hiểu hơn về những nội dung cơ bản của bài toán Nhận diện biển báo và đèn tín hiệu giao thông. Tìm hiểu được thêm nhiều khái niệm, nội dung cơ bản liên quan đến đề tài như những thuật ngữ cơ bản trong nghành trí tuệ nhân tạo, học máy, nhận dạng vật thể, những khái niệm về phương pháp học sâu sử dụng mạng neural để nhận diện vật thể... Cài đặt và áp dụng chạy thử mô hình đối với bài toán đặt ra. Sau quá trình xử lý và điều chỉnh dữ liệu, môi trường huấn luyện, thuật toán cũng đã cho ra những kết quả khả quan, khả thi trong việc sử dụng vào thực tế. Tuy nhiên em vẫn gặp phải nhiều khó khăn và hạn chế trong quá trình thực hiện.

Khó khăn của bản thân là chưa có nhiều kiến thức, kinh nghiệm thực tế để xây dựng một mô hình tốt hơn có thể giải quyết bài toán nêu trên. Do đây cũng là lần đầu tiên em tham gia dự án về trí tuệ nhân tạo, cụ thể là Thị giác máy tính, vẫn còn nhiều vấn đề em chưa nắm rõ và hiểu hết được. Đồng thời trong quá trình xây dựng và huấn luyện model, các thông số kỹ thuật vẫn chưa được như mong muốn. Vì vậy vẫn rất cần cải tiến hơn trong giai đoạn sau này..

5.2 Hướng phát triển trong tương lai

Dựa vào sự phát triển công nghệ mạnh mẽ như hiện nay thì tương lai có thể sử dụng các phiên bản YOLO mới nhất và nhiều mạng khác như Single Shot Detector, RentiaNet, CenterNet,... để có thể huấn luyện và nhận dạng được nhiều nhóm biển báo giao thông một cách chính xác và nhanh gọn hơn nhằm:

- Cải tiến chất lượng bộ huấn luyện phát hiện ảnh biển báo.
- Mở rộng cơ sở dữ liệu biển báo giao thông.
- Cải tiến phương pháp giải quyết trường hợp các biển báo bị hư hỏng hoặc bị chồng lấp.
- Nâng cấp và hoàn thiện khả năng của hệ thống trở thành một hệ thống nhận dạng và đưa ra cảnh báo tức thời cho người tham gia giao thông trong một chương trình hoàn chỉnh.

TÀI LIỆU THAM KHẢO

- [1] Afshine Amidi, Shervine Amidi, *Convolutional neural networks cheatsheet*.
url: <https://stanford.edu/~shervine/teaching/cs-230/cheatsheet-convolutional-neural-networks> (**urlseen** 22/07/2022).
- [2] Rohith Gandhi, *R-cnn, fast r-cnn, faster r-cnn, yolo — object detection algorithms*.
url: <https://towardsdatascience.com/r-cnn-fast-r-cnn-faster-r-cnn-yolo-object-detection-algorithms-36d53571365e> (**urlseen** 25/07/2022).
- [3] Jędrzej Świeżewski, Ph.D., *Yolo algorithm and yolo object detection*. **url:** <https://appsilon.com/object-detection-yolo-algorithm/> (**urlseen** 22/07/2022).