

TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI
VIỆN TOÁN ỨNG DỤNG VÀ TIN HỌC



ỨNG DỤNG HỌC TÍCH CỰC
CHO BÀI TOÁN PHÁT HIỆN VŨ KHÍ

ĐỒ ÁN III

Chuyên ngành: Toán Tin

Chuyên sâu: Tin học

Giảng viên hướng dẫn: TS. LÊ HẢI HÀ

Sinh viên thực hiện: NGUYỄN ĐỨC THẮNG

Lớp: KSTN Toán Tin - K61

HÀ NỘI, 01/2021

NHẬN XÉT CỦA GIẢNG VIÊN HƯỚNG DẪN

1. Mục đích và nội dung của đồ án:

2. Kết quả đạt được:

3. Ý thức làm việc của sinh viên:

Hà Nội, ngày tháng 01 năm 2021

Giảng viên hướng dẫn
(Ký và ghi rõ họ tên)

Mục lục

1	Tổng quan về bài toán	2
2	Các kiến thức cơ sở	5
2.1	Convolution neural network	5
2.1.1	Cấu trúc mạng CNN	5
2.1.2	Lớp tích chập	6
2.1.3	Padding	7
2.1.4	Lớp pooling	8
2.1.5	Lớp kết nối dày đặc	9
2.2	Transfer learning	10
2.2.1	Feature extractor	10
2.2.2	Fine tuning	11
2.3	Các độ đo đánh giá	13
2.3.1	Precision	13
2.3.2	Recall	13
2.3.3	IoU (Intersection over Union)	14
2.3.4	mAP (mean Average Precision)	14
3	Hệ mô hình YOLO	18
3.1	YOLO phiên bản 1	18
3.2	YOLO phiên bản 2	23
3.3	YOLO phiên bản 3	28
3.4	YOLO phiên bản 4	31
4	Tổng quan về học tích cực	34
4.1	Giới thiệu về học tích cực	34
4.1.1	Khái niệm học tích cực	34
4.1.2	Ví dụ về học tích cực	35
4.2	Kịch bản	36
4.2.1	Membership query synthesis	36
4.2.2	Stream-based selective sampling	37
4.2.3	Pool-based sampling	38
4.3	Các chiến lược truy vấn trong học tích cực	39

4.3.1	Lấy mẫu không chắc chắn	39
4.3.2	Truy vấn dựa vào hội đồng	41
5	Ứng dụng học tích cực cho bài toán phát hiện đối tượng	43
5.1	Học tích cực trong bài toán phát hiện đối tượng	43
5.2	Thu thập và biểu diễn dữ liệu	45
5.3	Cấu hình YOLO	47
5.4	Kết quả thực nghiệm	48

Danh sách hình vẽ

1.1	Một số bài toán trong thị giác máy tính	3
1.2	Phát hiện súng trong ảnh	4
2.1	Cấu trúc mạng CNN	6
2.2	Ví dụ về phép tích chập với kernel/filter 3×3 trên ảnh	7
2.3	Kích thước ảnh giảm nhanh chóng khi dùng tích chập	7
2.4	Thêm padding vào ảnh	8
2.5	Minh họa cho stride	8
2.6	Hoạt động của pooling	9
2.7	Làm phẳng đầu vào ảnh	9
2.8	Bên trái là mô hình VGG16, bên phải là mô hình VGG16 chỉ bao gồm ConvNet với đầu ra là các đặc trưng - Feature Extractor.	10
2.9	Phân loại chiến lược fine tuning	12
2.10	Intersection over Union	14
2.11	Dánh giá độ tốt của hộp giới hạn trên IoU. Màu xanh biểu thị hộp giới hạn thật của đối tượng, màu đỏ biểu thị hộp giới hạn dự đoán.	14
2.12	Ví dụ về mối quan hệ giữa recall và precision khi ngữ cảnh giảm dần . .	15
2.13	Thay giá trị precision bằng giá trị precision lớn nhất tại recall level .	15
2.14	Minh họa tính AP theo 11 points interpolated	16
2.15	Minh họa tính AP theo all points interpolated	16
3.1	Hệ thống phát hiện YOLO, (1) Ảnh được resize sang kích cỡ 448x448, (2) Chạy một mạng CNN đơn giản trên ảnh, (3) Phát hiện ngữ cảnh kết quả bằng đánh giá của mô hình.	18
3.2	Hệ thống lưới của YOLO	19
3.3	Ví dụ về kết quả đầu ra của YOLO khi nbox=2, nclass=20	20
3.4	Ô vuông trong hệ thống lưới	20
3.5	Dự đoán 2 hộp giới hạn cho mỗi ô vuông	20
3.6	Kiến trúc mạng YOLOv1	21
3.7	YOLOv2 dự đoán cho 5 anchor box tại mỗi ô lưới	24
3.8	Dự đoán hộp giới hạn trong YOLOv2	25
3.9	Kiến trúc Yolov2	25
3.10	Kỹ thuật Reorg trong Yolov2	26

3.11	Kiến trúc Darnknet-19	27
3.12	WordTree - YOLO9000	27
3.13	Kiến trúc Darnknet-53	29
3.14	Kiến trúc YOLOv3	30
3.15	Kiến trúc chung cho các trình phát hiện đối tượng	32
4.1	Ví dụ về học tích cực	35
4.2	Sơ đồ minh họa ba kịch bản học tích cực chính	36
4.3	Sơ đồ membership query synthesis	37
4.4	Sơ đồ stream-base selective sampling	38
4.5	Sơ đồ pool-base sampling	39
4.6	Heatmap minh họa hành vi truy vấn của các phương pháp lấy mẫu không chắc chắn phổ biến trong một bài toán phân loại ba nhãn.	40
5.1	Một mẫu ảnh trong bộ dữ liệu	45
5.2	Nhãn tương ứng với mẫu ảnh hình 5.1	46
5.3	Tương quan của nhãn dữ liệu	46
5.4	Cấu hình cho mô hình YOLO	47
5.5	Dồ thị biểu diễn độ đo precision trên bộ dữ liệu kiểm thử	48
5.6	Dồ thị biểu diễn độ đo recall trên bộ dữ liệu kiểm thử	49
5.7	Dồ thị biểu diễn độ đo mAP@0.5 trên bộ dữ liệu kiểm thử	49
5.8	Dồ thị biểu diễn độ đo mAP@[0.5:0.05:0.95] trên bộ dữ liệu kiểm thử	50
5.9	Sai số dự đoán đối tượng	50
5.10	Sai số phân lớp. Do chỉ sử dụng một lớp dữ liệu nên sai số này luôn bằng 0.	51
5.11	Sai số định vị hộp giới hạn	51
5.12	Một số kết quả dự đoán	52

Lời mở đầu

Ngày nay, an ninh ngày càng trở nên quan trọng với từng cá nhân hay các quốc gia. Hàng năm, có rất nhiều vụ khủng bố, cướp ngân hàng, ... mà người vi phạm sử dụng vũ khí để tấn công, đe doạ, gây nguy hại cho xã hội và cộng đồng; đặc biệt là các quốc gia cho phép sử dụng vũ khí. Quản lý, theo dõi hành vi cũng như phát hiện sử dụng vũ khí trái phép là một trong những vấn đề nhức nhối mà mỗi quốc gia, cá nhân đang nghiên cứu và phát triển. Tất nhiên, với thời đại phát triển công nghiệp 4.0 hiện nay, lượng dữ liệu ngày càng tăng cao nhưng nhân lực chưa đáp ứng được nhu cầu gán nhãn dữ liệu. Các hành vi của tội phạm ngày càng lưu manh và đa dạng hơn, đòi hỏi các mô hình phải "học" cách thay đổi thích ứng nhanh nhất giúp việc ngăn ngừa vũ khí trái phép nhanh nhất và hiệu quả.

Từ những đặc điểm trên, đồ án này nghiên cứu về học tích cực và ứng dụng cho bài toán phát hiện súng ngắn trong hình ảnh. Đồ án hoàn toàn có thể mở rộng tổng quát cho các đối tượng khác cũng như nhiều đối tượng. Đồ án này được chia làm 5 chương:

- **Chương 1:** Tổng quan về bài toán - Giới thiệu vấn đề, bài toán cần giải quyết.
- **Chương 2:** Các kiến thức cơ sở - Trình bày các kiến thức nền tảng cho nội dung của đồ án.
- **Chương 3:** Họ mô hình YOLO - Họ mô hình phát hiện đối tượng nhanh và hiệu quả, đây là mô hình được thử nghiệm và lựa chọn làm nhân của học tích cực trong phát hiện đối tượng.
- **Chương 4:** Tổng quan về học tích cực - Giới thiệu tổng quan, các phương pháp truy vấn trong học tích cực.
- **Chương 5:** Ứng dụng học tích cực cho bài toán phát hiện đối tượng - Nêu thuật toán học tích cực cho bài toán phát hiện đối tượng, mô tả dữ liệu và xử lý dữ liệu, kết quả thử nghiệm.

Chân thành cảm ơn TS. Lê Hải Hà và công ty Skymap Việt Nam đã hướng dẫn em trong suốt quá trình làm đồ án.

Với thời gian hạn hẹp, cũng như kiến thức giới hạn. Đồ án không thể tránh khỏi những thiếu sót, em rất mong nhận được sự góp ý của thầy cô và bạn bè để báo cáo được hoàn thiện hơn.

Chương 1

Tổng quan về bài toán

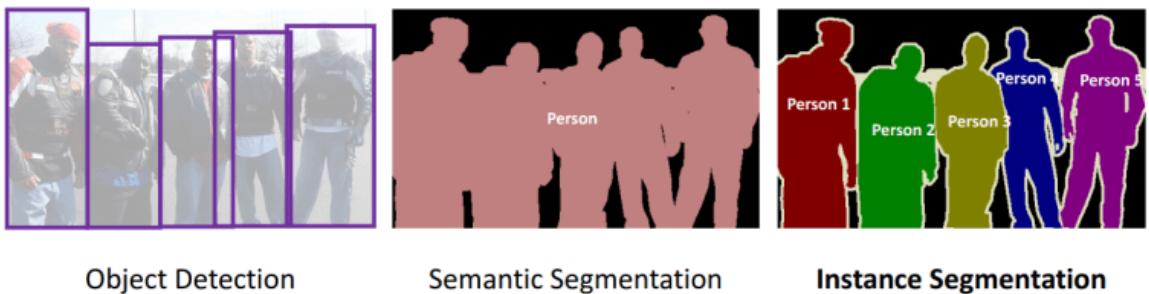
Trí tuệ nhân tạo là một trong những ngành chủ chốt cho thời đại công nghiệp 4.0. Trong đó, học máy (machine learning) là một phần của trí tuệ nhân tạo đang có những sự bùng nổ mạnh mẽ trong những năm gần đây. Các phương pháp học máy có thể được phân nhóm dựa trên phương thức học như sau:

- **Học có giám sát:** Thuật toán dự đoán đầu ra dựa trên bộ dữ liệu *có nhãn* đã biết trước. Đây là nhóm phổ biến nhất trong các thuật toán học máy. Ví dụ như bài toán chấm công bằng mặt người thì dữ liệu đầu vào cho phương pháp này là một cặp (ảnh mặt người, tên người).
- **Học không giám sát:** Trong nhóm phương pháp này, dữ liệu đầu vào không có nhãn, các thuật toán nhóm này dựa vào cấu trúc của dữ liệu để phân cụm, tìm ra quy luật.
- **Học bán giám sát:** Là các bài toán mà với một lượng lớn dữ liệu nhưng chỉ có một phần dữ liệu được gán nhãn. Phương pháp này được coi như là ở giữa 2 nhóm thuật toán trên.
- **Học tăng cường:** Là các phương pháp mà hệ thống tự xác định hành vi phù hợp nhất dựa trên ngữ cảnh để đạt được lợi ích cao nhất. Các thuật toán học tăng cường hiện tại chủ yếu áp dụng vào game, hệ thống sẽ tự động chơi game và tương tác môi trường để học cách xử lý thông minh hơn với từng tình huống.

Thị giác máy tính (Computer Vision) là một nhánh của trí tuệ nhân tạo, đang ngày càng phát triển trong những năm gần đây. Thị giác máy tính dần đi vào cuộc sống thực tế và có những bước tiến vượt trội với những bài toán có tính ứng dụng cao như: chấm công khuôn mặt trong các công ty, ô tô tự lái, camera an ninh, Trong thị giác máy tính chia thành các bài toán cơ bản sau:

- **Phân loại hình ảnh (image classification):** Liên quan đến việc gán nhãn, phân loại cho một 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 kết hợp cả hai nhiệm vụ trên. Vẽ một hộp giới hạn xung quanh từng đối tượng mà ta quan tâm trong ảnh và gán, phân loại cho chúng một nhãn.
- **Phân đoạn đối tượng (object segmentation):** Phân loại đối tượng trên pixel ảnh, xác định những pixel nào thuộc đối tượng. Trong phân đoạn đối tượng chia làm 2 loại là: semantic segmentation và instance segmentation. Trong đó semantic segmentation là thực hiện phân đoạn với từng lớp khác nhau, ví dụ tất cả người là một lớp, tất cả ô tô là một lớp. Instance segmentation là thực hiện phân đoạn với từng đối tượng trong một lớp. Ví dụ có 2 người trong ảnh thì phải phân biệt rõ ràng 2 vùng phân đoạn khác nhau cho mỗi người.
- **Chú thích ảnh (image captioning):** Dưa ra lý giải về hành động và nội dung bức ảnh hoặc một chuỗi bức ảnh, video.



Hình 1.1: Một số bài toán trong thị giác máy tính

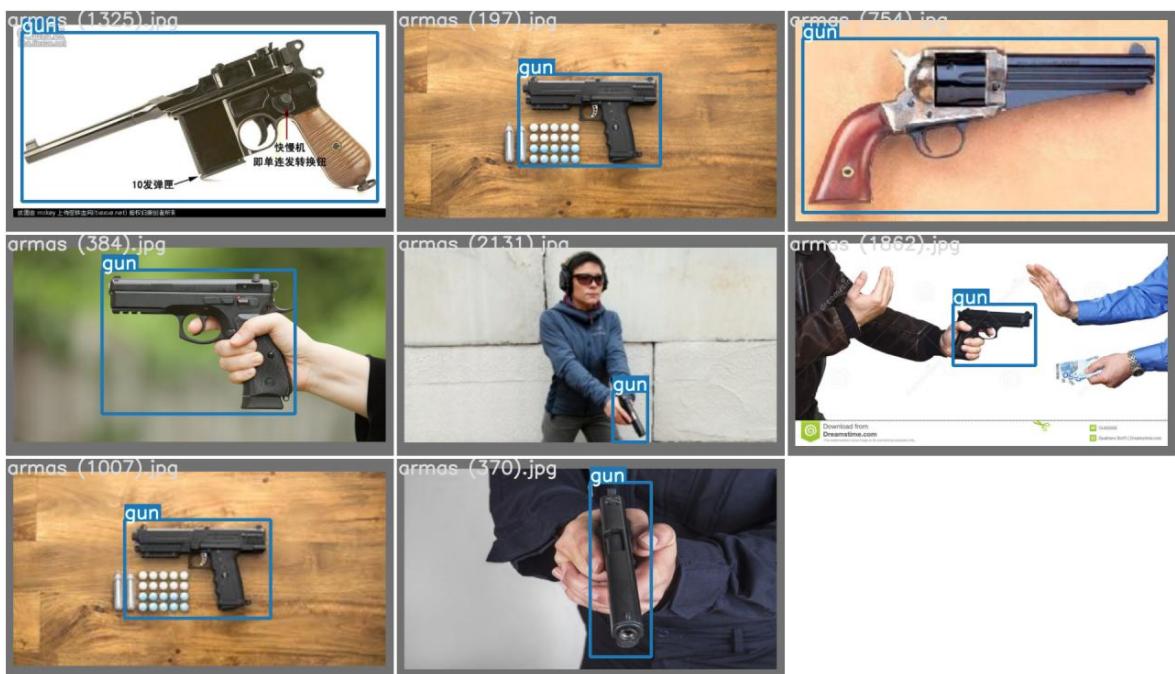
Các bài toán trong thị giác máy tính còn được tổ chức thành các cuộc thi với các giải thưởng cao, thúc đẩy nghiên cứu và phát triển. Thị giác máy tính đã đạt được nhiều thành tựu đột phá cả về tốc độ và độ chính xác trong những năm qua.

Với tốc độ phát triển hiện nay, dữ liệu ngày càng tăng. Tuy nhiên, lại chỉ có một lượng rất nhỏ dữ liệu có nhãn để phục vụ cho các bài toán. Nhân lực và tài nguyên gán nhãn chưa đáp ứng được nhu cầu xã hội và chưa đáp ứng được sự tăng trưởng mạnh mẽ của thị giác máy tính. Trong khi đó, những phương pháp như học không giám sát hay học tăng cường lại chưa phát triển đủ mạnh mẽ để có thể xử lý đa dạng nhiều bài toán.

Ngoài ra, các mô hình trích xuất đặc trưng cổ điển như HOG, SHIF, LBP chỉ lấy được những đặc trưng bề nổi của ảnh. Và vì thế, học sâu (deep learning) ngày càng phát triển để xử lý các bài toán của thị giác máy tính.

Với những vấn đề trên, đồ án này tập trung nghiên cứu một phương pháp học bán giám sát cho phát hiện đối tượng sử dụng học sâu. Đồ án tập trung vào việc chỉ sử dụng một phần dữ liệu được gán nhãn nhưng vẫn đảm bảo trình phát hiện đối tượng hoạt động tốt, giúp giảm chi phí và nhân lực cho việc gán nhãn dữ liệu.

Cụ thể, đồ án nghiên cứu phương pháp học tích cực cho bài toán phát hiện đối tượng. Với tình hình hiện nay, hàng năm có rất nhiều vụ khủng bố, cướp ngân hàng, sử dụng, tàng trữ vũ khí trái phép. Vì vậy, đồ án sử dụng học tích cực cho việc phát hiện đối tượng là vũ khí, cụ thể là súng, góp phần kiểm soát an ninh xã hội một cách tự động mà tốn ít chi phí gán nhãn nhất nhưng vẫn đem lại hiệu quả cao.



Hình 1.2: Phát hiện súng trong ảnh

Chương 2

Các kiến thức cơ sở

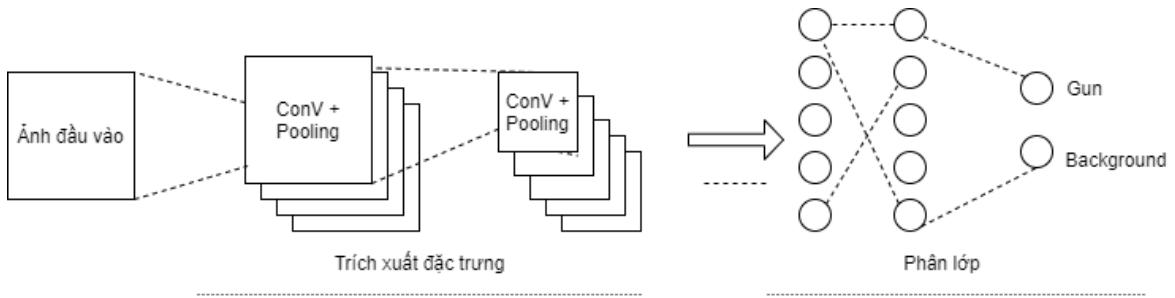
2.1 Convolution neural network

Convolutional Neural Network (CNN – Mạng neural tích chập) là một trong những mô hình học sâu tiên tiến. Nó giúp cho chúng ta xây dựng được những hệ thống thông minh với độ chính xác cao. CNN được sử dụng nhiều trong các bài toán nhận dạng các đối tượng trong ảnh.

2.1.1 Cấu trúc mạng CNN

Mạng CNN là một tập các lớp tích chập (convolution) chồng lên nhau và sử dụng các hàm phi tuyến như ReLU, tanh, sigmoid, ... là hàm kích hoạt. Càng những tầng sau của mạng CNN thì càng trích lọc những thông tin chi tiết và sâu hơn trong ảnh. Trong mô hình CNN, các lớp của mạng liên kết với nhau thông qua cơ chế tích chập. Lớp tiếp theo là kết quả tích chập từ lớp trước đó, nhờ vậy mà ta có được các kết nối cục bộ của các phần của ảnh với nhau.

Mạng CNN có kiến trúc khác với mạng neural thông thường. Mạng neural thông thường chuyển đầu vào thông qua hàng loạt tầng ẩn. Mỗi tầng là tập các neural và các tầng liên kết đầy đủ với nhau. Và tầng cuối cùng là tầng sẽ trả ra kết quả đại diện cho dự đoán của mạng. Tuy nhiên, mạng neural thông thường không đáp ứng được trong việc trích lọc đặc trưng ảnh vì nếu chỉ một ảnh đầu vào kích thước 3000×3000 pixel thì riêng tầng đầu vào của mạng neural đã cần 9000000 units. Nếu cứ xây dựng mạng neural cho xử lý ảnh sẽ tạo ra một khối lượng tính toán rất lớn. Dựa vào tính chất của ảnh, những pixel ở một cụm gần nhau có thể mang thông tin ý nghĩa như nhau, CNN ra đời giúp lấy thông tin cục bộ trong một vùng ảnh (subsampling) của ảnh nhưng khối lượng tính toán ít hơn.



Hình 2.1: Cấu trúc mạng CNN

Về cơ bản, mạng CNN gồm 2 thành phần:

- **Phần trích xuất đặc trưng:** Trong phần này, mạng sẽ tiến hành tính toán hàng loạt các phép tích chập (convolution) và phép hợp nhất (pooling) để phát hiện đặc trưng. Ví dụ: nếu ta có ảnh khẩu súng thì phần này sẽ nhận dạng đặc điểm chung và chi tiết của một khẩu súng như ngòi súng, màu súng, ...
- **Phần phân lớp:** Tại phần này, một lớp với các liên kết đầy đủ sẽ đóng vai trò như một bộ phân lớp các đặc trưng đã trích xuất được trước đó. Tầng này sẽ đưa ra xác suất của một đối tượng trong hình. Số units tầng cuối của tầng phân lớp phải bằng số lớp mà chúng ta cần phân loại. Ở hình 2.1, chúng ta phân loại thành 2 lớp đó là lớp ảnh súng và ảnh nền.

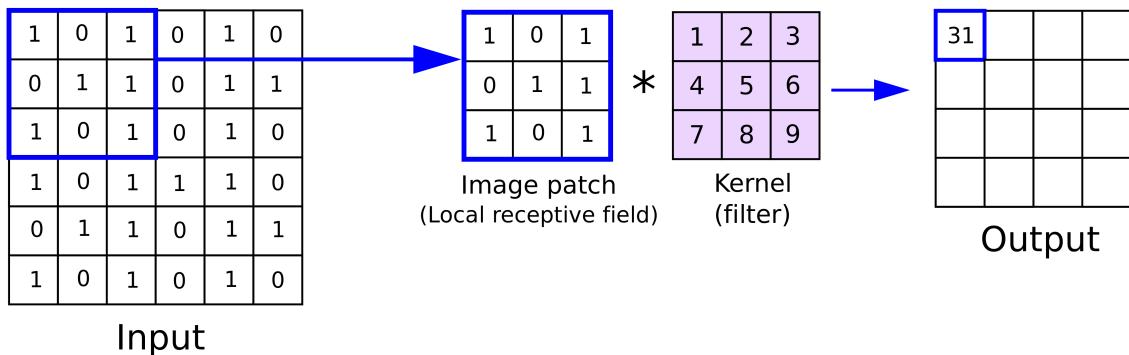
2.1.2 Lớp tích chập

Lớp tích chập (convolution layer) thường là lớp đầu tiên trong mô hình CNN. Lớp này có chức năng phát hiện ra các đặc trưng về không gian một cách hiệu quả.

Filter

Filter (hay còn gọi là kernel) là một trong những tham số quan trọng trong CNN. Kích thước filter trong tầng tích chập phổ biến hiện nay là 3×3 . Kích thước filter thường là số lẻ, ví dụ $3 \times 3, 5 \times 5, 7 \times 7$.

Kích thước filter thường không quá lớn. Vì kích thước nhỏ nó có thể trích xuất cục bộ chi tiết hơn, kích thước ảnh giảm chậm hơn. Ta thực hiện các phép tích chập bằng cách trượt filter từ trái sang phải, từ trên xuống dưới theo dữ liệu đầu vào. Tại mỗi vị trí, ta tiến hành phép nhân ma trận và tính tổng tất cả các giá trị để đưa vào bản đồ đặc trưng.



Hình 2.2: Ví dụ về phép tích chập với kernel/filter 3×3 trên ảnh

Với ảnh RGB có 3 kênh màu red, green, blue thì filter phải có cùng độ sâu (depth) với ảnh.

2.1.3 Padding

Khi dùng các phép tích chập, thông tin ở biên bức ảnh bị biến mất và kích thước của ảnh giảm nhanh chóng (Hình 2.3).

Ảnh 3x3	Filter 3x3	Feature map 1x1
$\begin{array}{ c c c } \hline 0 & 4 & 2 \\ \hline 1 & 1 & 0 \\ \hline 3 & 4 & 5 \\ \hline \end{array}$	$*$ $\begin{array}{ c c c } \hline 0 & 1 & 0 \\ \hline 0 & 1 & 0 \\ \hline 0 & 1 & 0 \\ \hline \end{array}$	$=$ $\boxed{9}$

Hình 2.3: Kích thước ảnh giảm nhanh chóng khi dùng tích chập

Để khắc phục vấn đề này, chúng ta sử dụng padding. Bằng việc thêm các giá trị 0 vào biên, ta sẽ có *zero padding* (Hình 2.4).

Ảnh 3x3 được thêm pad	Filter 3x3	Feature map 3x3																																											
<table border="1" style="border-collapse: collapse; width: 100%;"> <tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td></tr> <tr><td>0</td><td>0</td><td>4</td><td>2</td><td>0</td></tr> <tr><td>0</td><td>1</td><td>1</td><td>0</td><td>0</td></tr> <tr><td>0</td><td>3</td><td>4</td><td>5</td><td>0</td></tr> <tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td></tr> </table>	0	0	0	0	0	0	0	4	2	0	0	1	1	0	0	0	3	4	5	0	0	0	0	0	0	$*$ <table border="1" style="border-collapse: collapse; width: 100%;"> <tr><td>0</td><td>1</td><td>0</td></tr> <tr><td>0</td><td>1</td><td>0</td></tr> <tr><td>0</td><td>1</td><td>0</td></tr> </table>	0	1	0	0	1	0	0	1	0	$=$ <table border="1" style="border-collapse: collapse; width: 100%;"> <tr><td>1</td><td>5</td><td>2</td></tr> <tr><td>4</td><td>9</td><td>7</td></tr> <tr><td>4</td><td>5</td><td>6</td></tr> </table>	1	5	2	4	9	7	4	5	6
0	0	0	0	0																																									
0	0	4	2	0																																									
0	1	1	0	0																																									
0	3	4	5	0																																									
0	0	0	0	0																																									
0	1	0																																											
0	1	0																																											
0	1	0																																											
1	5	2																																											
4	9	7																																											
4	5	6																																											

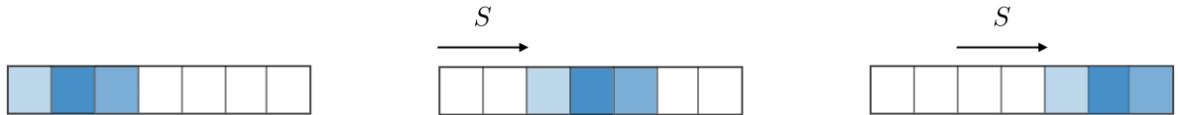
Hình 2.4: Thêm padding vào ảnh

Sau khi thêm padding chúng ta có một số lợi thế như sau:

- Không mất mát thông tin viền nên nhận diện sẽ tốt hơn, tìm được chính xác đối tượng hơn.
- Đầu ra của CNN kích thước sẽ giảm dần nên khi thêm padding sẽ giúp giảm chậm hơn.

Stride

Dối với phép convolution hay pooling thì stride (S) là độ dài bước trượt của filter. Như trong hình 2.5, độ dài bước trượt $S = 2$.



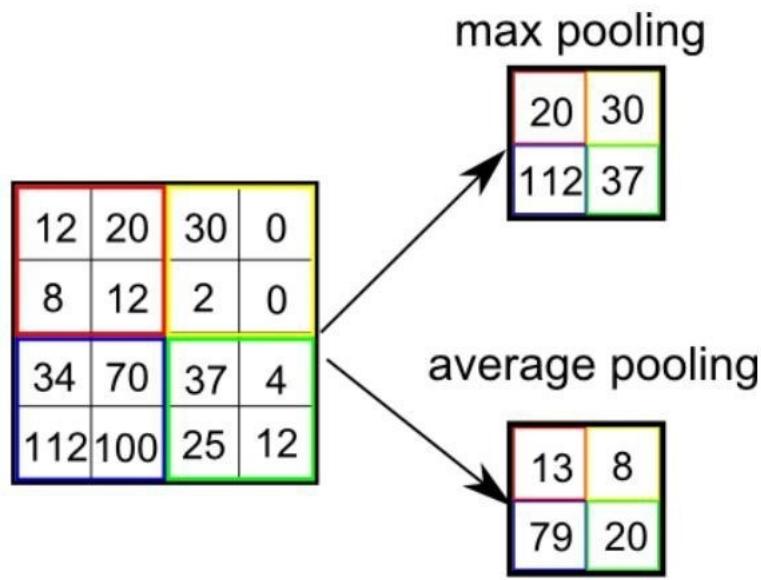
Hình 2.5: Minh họa cho stride

2.1.4 Lớp pooling

Lớp pooling thường được dùng giữa các lớp tích chập, để giảm kích thước dữ liệu nhưng vẫn giữ được các thuộc tính quan trọng. Kích thước dữ liệu giảm giúp giảm việc tính toán trong mô hình.

Có 2 loại pooling thường được sử dụng trong CNN:

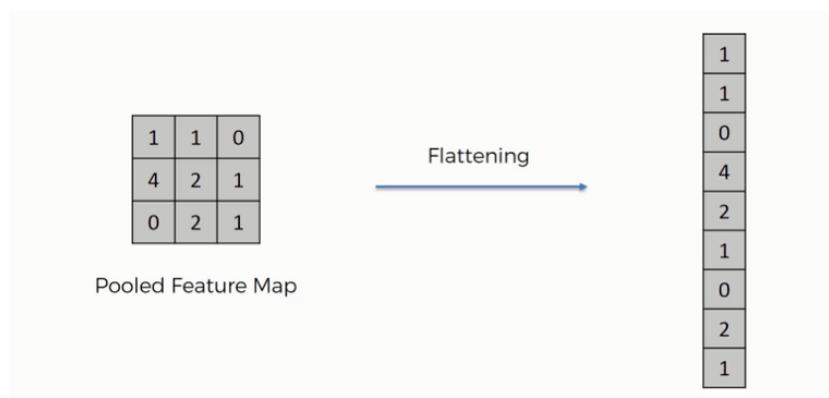
- **Max Pooling:** Thực hiện lấy giá trị lớn nhất trong kích thước filter mà ta xét.
- **Average Pooling:** Thực hiện lấy giá trị trung bình tổng trong kích thước filter ta xét.



Hình 2.6: Hoạt động của pooling

2.1.5 Lớp kết nối dày đặc

Trong phần phân lớp, ta sử dụng một vài tầng với kết nối dày đủ - lớp kết nối dày đặc (fully connected layer) để xử lý kết quả của phần tích chập. Vì đầu vào của mạng liên kết dày đủ là 1 chiều, ta cần làm phẳng đầu vào trước khi phân lớp (Hình 2.7). Tầng cuối cùng trong mạng CNN là một tầng liên kết dày đủ, phần này hoạt động tương tự như mạng neural thông thường.



Hình 2.7: Làm phẳng đầu vào ảnh

Kết quả thu được cuối cùng sẽ là một vector với các giá trị xác suất cho việc dự đoán như mạng neural thông thường.

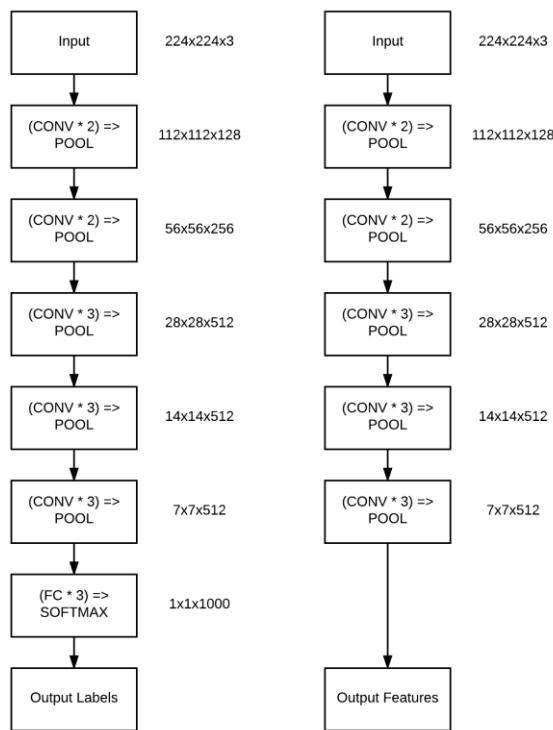
2.2 Transfer learning

Transfer learning là cách để các mô hình truyền đạt cho nhau khả năng mà mỗi mô hình có thể làm được. Trong transfer learning, một mô hình đã huấn luyện (pretrained model) trên một bài toán cụ thể nào đó thực hiện một nhiệm vụ hoặc nhiều nhiệm vụ (gọi là *source tasks*). Mô hình mới sử dụng một phần hay toàn bộ pretrained model để học một bài toán khác có thể cùng mục đích hoặc không cùng mục đích với pretrained model đã được huấn luyện (gọi là *target tasks*). Tuỳ vào nhiệm vụ của mỗi layer mà mô hình mới có thể thêm các layer khác dựa trên pretrained model sẵn có.

Có 2 loại transfer learning: Feature extractor và Fine tuning

2.2.1 Feature extractor

Feature extractor là một phần của mô hình dùng để trích xuất ra đặc trưng nói chung. Chúng ta có thể sử dụng feature extractor đã được huấn luyện để trích xuất ra các đặc trưng của mô hình thay vì phải tạo ra một trình trích xuất đặc trưng mới và huấn luyện lại từ đầu. Những đặc trưng này sau đó được dùng làm đầu vào của những thuật toán học máy để học và dự đoán các kết quả từ những đặc trưng được tạo ra ở bước trên.



Hình 2.8: Bên trái là mô hình VGG16, bên phải là mô hình VGG16 chỉ bao gồm ConvNet với đầu ra là các đặc trưng - Feature Extractor.

2.2.2 Fine tuning

Để sử dụng mô hình pretrained hiệu quả, chúng ta cần:

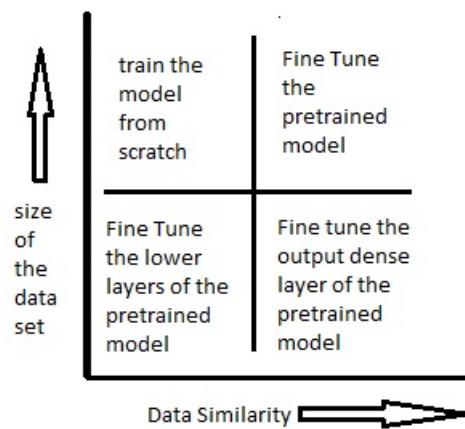
- Thêm các tầng phù hợp với bài toán của chúng ta, loại bỏ các tầng của mô hình pretrained mà chúng ta không dùng đến, đây là một vấn đề khó cần phải có những nghiên cứu chuyên sâu về từng tầng và mục đích của chúng.
- Có chiến lược huấn luyện tốt, vì nếu huấn luyện không tốt sẽ làm mất đi tính hiệu quả của mô hình pretrained và do đó giảm khả năng của mô hình.

Do đó, *fine tuning* ra đời nhằm giúp chúng ta có chiến lược huấn luyện hiệu quả. Fine tuning là việc huấn luyện một mô hình transferred nhằm mục đích tối ưu độ chính xác của mô hình trên mục tiêu bài toán.

Dưới đây là một số chiến lược fine tuning thường dùng (hình 2.9):

- Khi dữ liệu cho mục tiêu (*target task*) *lớn và tương tự* với dữ liệu nguồn (*source tasks*): Đây là trường hợp lý tưởng, có thể dùng các trọng số (*weights*) của mô hình pretrained để khởi tạo cho phần pretrained.
- Khi dữ liệu cho mục tiêu *nhỏ và tương tự* với dữ liệu nguồn: Vì dữ liệu nhỏ, nếu huấn luyện lại phần pretrained sẽ dẫn đến mô hình quá khớp (*overfitting*), do đó chúng ta chỉ huấn luyện những tầng được thêm vào với phần trọng số khởi tạo cho pretrained như trên.
- Khi dữ liệu cho mục tiêu *lớn và khác biệt* với dữ liệu nguồn: bởi vì dữ liệu của chúng ta có sự khác biệt nên khi dùng trọng số từ mô hình pretrained sẽ làm giảm độ chính xác vì sự khác biệt trong tác vụ và dữ liệu, nhưng cũng chính vì dữ liệu lớn nên việc huấn luyện toàn bộ mô hình transferred từ đầu là hiệu quả nhất, giúp cho mô hình thích nghi tốt hơn với dữ liệu này.
- Khi dữ liệu cho mục tiêu *nhỏ và khác biệt* với dữ liệu nguồn: đây là trường hợp khó khăn nhất, chúng ta có thể can thiệp vào mô hình pretrained, thay thế những tầng pretrained xa đầu vào để thích nghi với dữ liệu mới nhưng không được huấn luyện các tầng gần đầu vào của pretrained vì dữ liệu nhỏ sẽ không thể huấn luyện được các tầng này hiệu quả và các tầng này chỉ trích xuất các đặc trưng tổng quát từ dữ liệu, sẽ không ảnh hưởng đến bài toán mục tiêu.

Transfer learning mang đến những mô hình mới với độ chính xác cao trong thời gian ngắn. Tuy nhiên, nó cũng không phải là một kỹ thuật dễ sử dụng. Nếu sai sót thì có thể gây ra kết quả tệ hơn cả ban đầu. Hầu hết các mô hình dùng transfer learning được sử dụng trong các nghiên cứu về thị giác máy tính, chú trọng vào việc trích xuất các đặc trưng từ ảnh hoặc video một cách hiệu quả như một cách thay thế cho các phương pháp cũ và kết hợp những ý tưởng mới để tận dụng các đặc trưng này.



Hình 2.9: Phân loại chiến lược fine tuning

Transfer learning cũng được sử dụng rất nhiều trong xử lý ngôn ngữ tự nhiên - natural language processing (NLP). Trên thực tế thì: nếu thị giác máy tính dùng mạng tích chập để trích xuất các đặc trưng từ ảnh thì NLP dùng word embedding như một cách để trích xuất các đặc trưng từ các từ thành những vector. Hiệu quả thực tiễn của word embedding cao hơn hẳn one-hot encoding về khả năng biểu diễn thông tin.

2.3 Các độ đo đánh giá

Khi xây dựng mô hình học máy, chúng ta cần các phương pháp đánh giá để xem mô hình có hiệu quả hay không. Có rất nhiều phương pháp đánh giá mô hình, tùy thuộc vào bài toán mà chúng ta có những độ đo đánh giá khác nhau. Trong phần này sẽ giới thiệu một số phương pháp, độ đo để đánh giá mô hình phù hợp với bài toán phát hiện đối tượng trong ảnh.

Một số ký hiệu:

- **TP**: True positive - Các vùng chứa đối tượng trong ảnh và mô hình nhận dạng đúng.
- **FP**: False positive - Các vùng không chứa đối tượng trong ảnh, nhưng mô hình lại nhận dạng là chứa đối tượng.
- **TN**: True negative - Các vùng không chứa đối tượng trong ảnh và mô hình nhận dạng đúng.
- **FN**: False negative - Các vùng chứa đối tượng trong ảnh, nhưng mô hình lại nhận dạng là không có đối tượng.

2.3.1 Precision

Precision được định nghĩa là tỉ lệ số điểm **true positive** trong số những điểm được phân loại là **positive** (**TP + FP**). Precision cao đồng nghĩa với việc độ chính xác của các điểm tìm được là cao.

Với precision, chúng ta có công thức mô tả như sau:

$$\text{precision} = \frac{\text{TP}}{\text{TP}+\text{FP}}$$

2.3.2 Recall

Recall được định nghĩa là tỉ lệ số điểm **true positive** trong số những điểm thực sự là **positive** (**TP + FN**). Recall dại diện cho độ bao phủ, recall cao có nghĩa tỉ lệ bỏ sót các sample positive thực thấp.

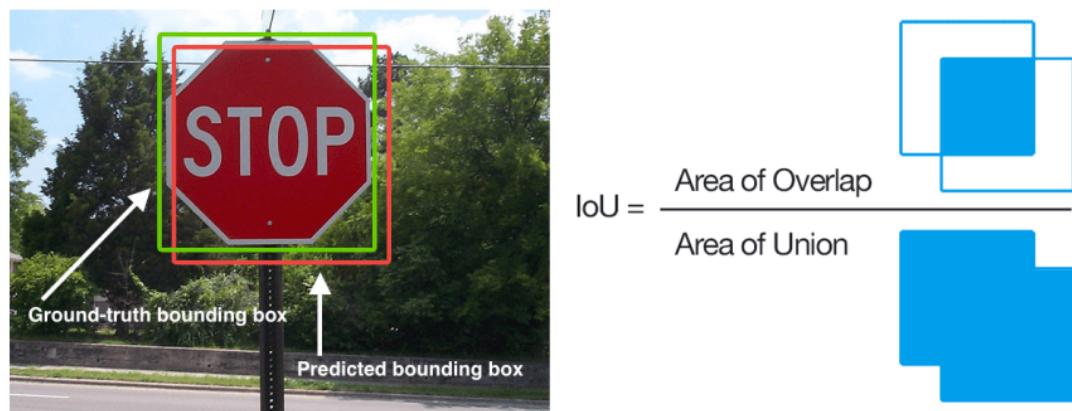
Với Recall, chúng ta có công thức mô tả như sau:

$$\text{Recall} = \frac{\text{TP}}{\text{TP}+\text{FN}}$$

Dể có thể sử dụng precision và recall cho bài toán phân lớp nhiều đối tượng hay phát hiện nhiều lớp đối tượng. Ta có thể đưa bài toán phân lớp nhiều lớp về bài toán phân lớp nhị phân bằng cách xem xét từng lớp. Với mỗi lớp, ta coi dữ liệu thuộc lớp đó có nhãn là positive, tất cả các dữ liệu còn lại có nhãn là negative. Sau đó, giá trị precision, recall được áp dụng lên từng lớp.

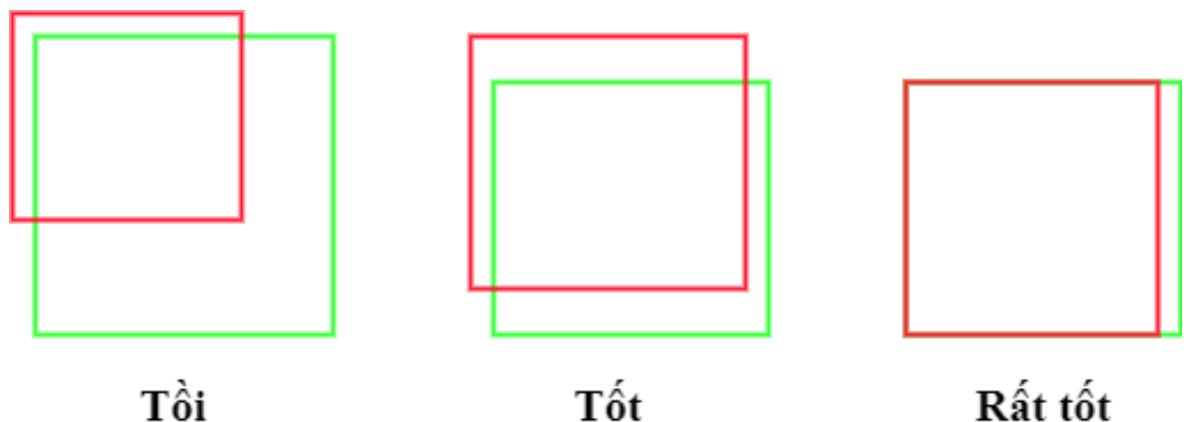
2.3.3 IoU (Intersection over Union)

Intersection over Union (IoU) được sử dụng trong bài toán phát hiện đối tượng, để đánh giá xem hộp giới hạn dự đoán đối tượng khớp với hộp giới hạn thật của đối tượng. Ta sẽ đánh giá mô hình bằng tỉ lệ vùng giao nhau (area overlap) với vùng hợp nhau (area union) giữa thực tế và dự đoán. Hình 2.10 cho thấy các tính IoU.



Hình 2.10: Intersection over Union

Dễ thấy rằng, chỉ số IoU sẽ nằm trong khoảng $[0, 1]$ và IoU càng gần 1 thì hộp giới hạn dự đoán càng gần với hộp giới hạn thật.

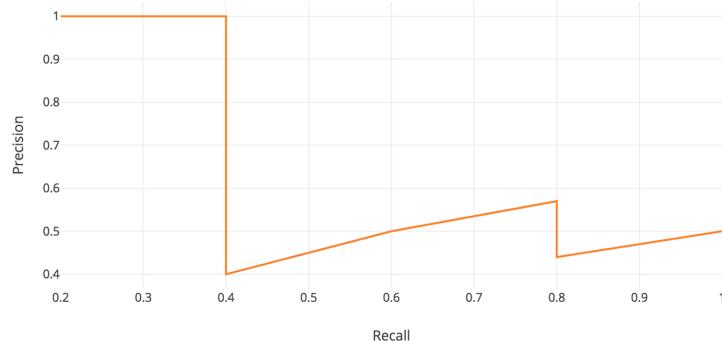


Hình 2.11: Đánh giá độ tốt của hộp giới hạn trên IoU. Màu xanh biểu thị hộp giới hạn thật của đối tượng, màu đỏ biểu thị hộp giới hạn dự đoán.

2.3.4 mAP (mean Average Precision)

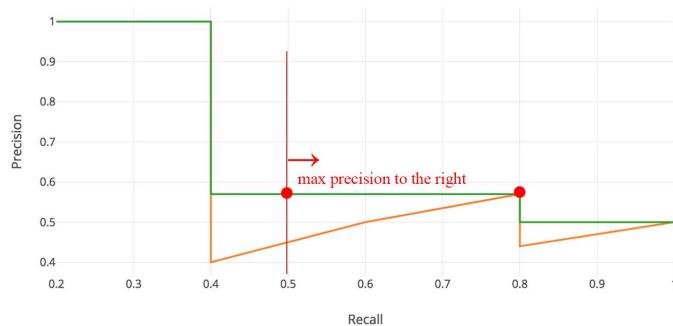
Với bài toán phát hiện đối tượng, việc định nghĩa TP, FP, TN, FN phụ thuộc vào ngưỡng chọn IoU. Khi IoU thay đổi thì việc đánh giá precision và recall cũng thay đổi. **mAP** là một độ đo tốt để đánh giá bài toán phát hiện đối tượng.

Khi càng giảm ngưỡng quyết định thì recall tăng, còn precision thay đổi không ổn định. Hình 2.12 mô tả một ví dụ cho mối quan hệ này.



Hình 2.12: Ví dụ về mối quan hệ giữa recall và precision khi ngưỡng giảm dần

Giá trị AP là giá trị phía dưới đường biểu diễn mối quan hệ precision – recall. Để đơn giản cho việc tính toán, tại mỗi recall level, ta thay giá trị precision bằng giá trị precision lớn nhất tại recall level đó. Có 2 cách tính toán là **11 points interpolated** và **all points interpolated**.



Hình 2.13: Thay giá trị precision bằng giá trị precision lớn nhất tại recall level

11 points interpolated

Chia đường biểu diễn precision - recall (sau khi đã xấp xỉ) thành 11 đoạn bằng nhau theo recall ($0, 0.1, 0.2, \dots, 1$). Khi đó, AP được tính bằng công thức:

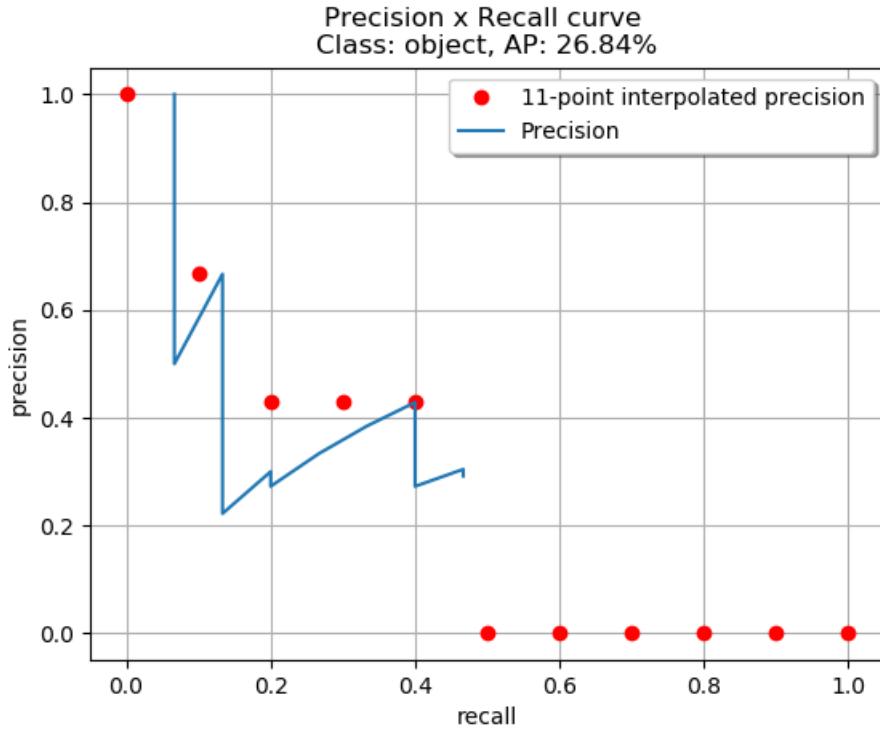
$$AP = \frac{1}{11} \times (AP_r(0) + AP_r(0.1) + \dots + AP_r(1.0))$$

hay:

$$AP = \frac{1}{11} \sum_{r \in \{0, 0.1, \dots, 1.0\}} p_{\text{interp}}(r)$$

trong đó:

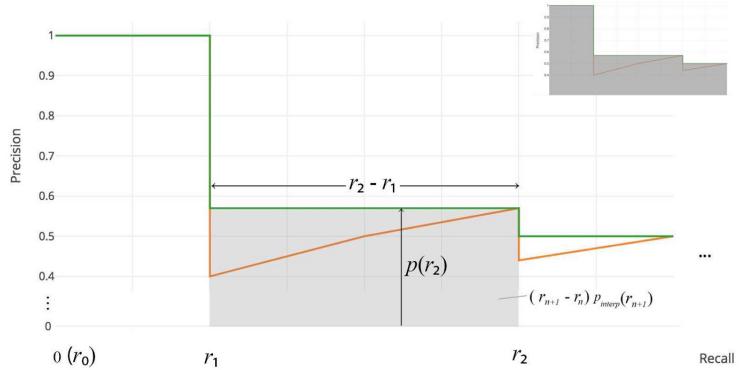
$$p_{\text{interp}}(r) = \max_{\tilde{r} \geq r} p(\tilde{r})$$



Hình 2.14: Minh họa tính AP theo 11 points interpolated

all points interpolated

Với all points interpolated, ta có thể tính luôn phần diện tích phía dưới đường biểu diễn precision-recall bằng cách tính tổng các hình chữ nhật xấp xỉ.



Hình 2.15: Minh họa tính AP theo all points interpolated

Chúng ta có công thức tính AP theo all points interpolated như sau:

$$AP = \sum (r_{n+1} - r_n) p_{\text{interp}}(r_{n+1})$$

trong đó:

$$p_{\text{interp}}(r_{n+1}) = \max_{\tilde{r} \geq r_{n+1}} p(\tilde{r})$$

Ta gọi:

- AP@[0.5:0.05:0.95]: Giá trị AP trung bình của các AP khi IOU có giá trị $(0.5, 0.55, \dots, 0.95)$
- AP@0.5: Giá trị AP khi IoU = 0.5

Sau khi tính các giá trị AP, thì **mAP** đơn giản là trung bình AP score của tất cả các lớp (giả sử có N lớp), được định nghĩa với công thức:

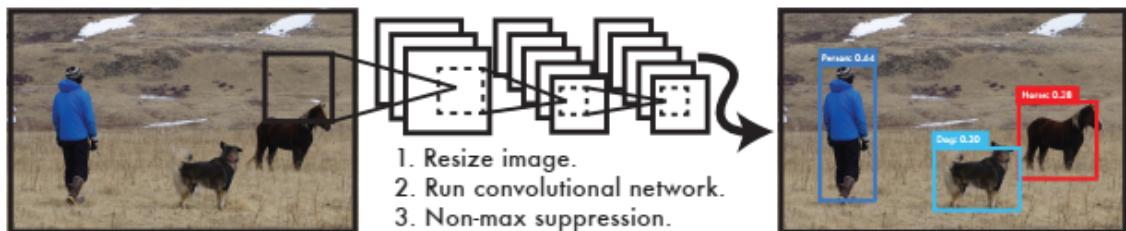
$$mAP = \frac{1}{N} \sum_{i=1}^N AP_i$$

Chương 3

Họ mô hình YOLO

3.1 YOLO phiên bản 1

YOLO phiên bản đầu tiên (YOLOv1) lần đầu được xuất bản bởi Joseph Redmon cùng các cộng sự vào năm 2016 trong paper **You Only Look Once: Unified, Real-Time Object Detection**. Các mô hình lúc bấy giờ sử dụng cửa sổ trượt để phát hiện đối tượng hoặc thuật toán selective search hoặc dùng mạng đề xuất khu vực (region proposal) để trích xuất ra các hộp giới hạn của các đối tượng trong ảnh và chạy trình phân lớp trên các hộp này. Sau khi phân lớp, hậu xử lý được sử dụng để tinh chỉnh các hộp giới hạn, loại bỏ các phát hiện trùng lặp, Do đó, quá trình này rất chậm và phức tạp, khó tối ưu hoá vì mỗi thành phần phải đào tạo riêng biệt.

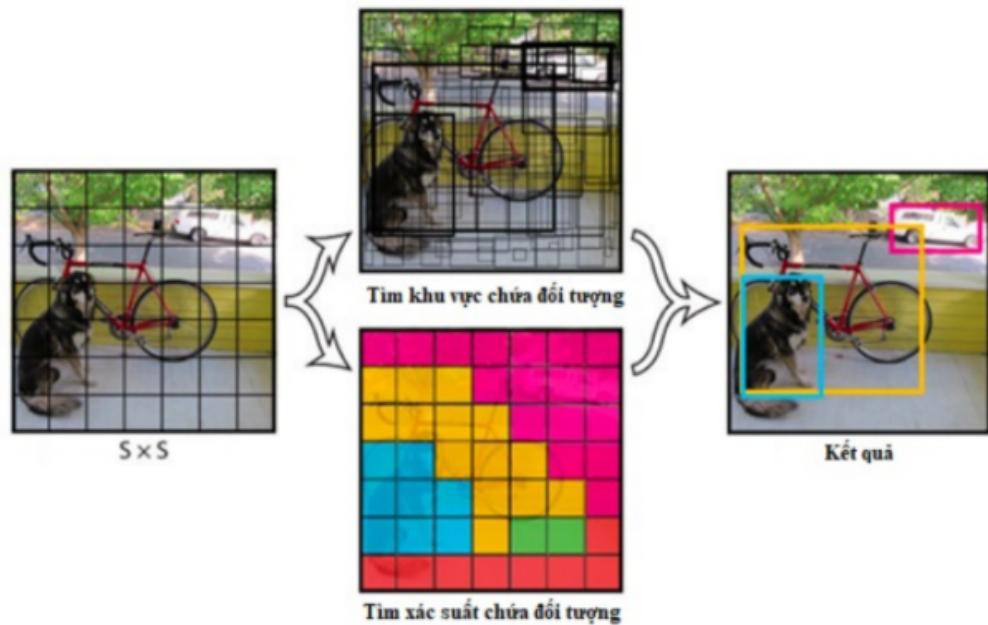


Hình 3.1: Hệ thống phát hiện YOLO, (1) Ảnh được resize sang kích cỡ 448x448, (2) Chạy một mạng CNN đơn giản trên ảnh, (3) Phát hiện ngưỡng kết quả bằng đánh giá của mô hình.

Các tác giả thống nhất các thành phần của phát hiện đối tượng thành một mạng neural duy nhất, sử dụng các đặc trưng từ toàn bộ hình ảnh để dự đoán các hộp giới hạn cho tất cả các lớp. Thiết kế YOLO cho phép huấn luyện end-to-end thời gian thực trong khi độ chính xác vẫn trung bình cao.

Hệ thống lưới

Trong mô hình YOLO, bức ảnh đầu vào sẽ được thay đổi kích thước thành một ảnh vuông (cụ thể với phiên bản YOLOv1 là 448×448). Trên ảnh vuông này, ta sẽ chia chúng thành một hệ thống lưới bao gồm các ô (Hình 3.2). Nếu trung tâm của đối tượng rơi vào một ô lưới đó, thì ô lưới đó có trách nhiệm phát hiện đối tượng đó.



Hình 3.2: Hệ thống lưới của YOLO

Các ô này sẽ làm hai nhiệm vụ đó là dự đoán xác suất chứa đối tượng trong đó và vị trí chứa đối tượng trong ô đấy. Vị trí của đối tượng ở đây được nói đến gồm 4 giá trị: vị trí tương đối của hoành độ (x) và tung độ (y) so với ô lưới, vị trí tương đối của chiều rộng (w) và chiều cao (h) so với toàn bộ bức ảnh. *Vị trí tương đối* a của x so với y là $a = x/y$.

Phụ thuộc vào việc muốn tăng độ chính xác cho mô hình, mỗi ô vuông có thể dự đoán nhiều hơn một hộp giới hạn. Thông thường, trong mô hình YOLOv1, số lượng hộp giới hạn mà mỗi ô vuông dự đoán là 2.

Giả sử ta muốn nhận diện nclass kiểu đối tượng, mỗi ô vuông ta muốn nhận nbox hộp giới hạn. Khi đó, vector đầu ra nhận được từ mỗi ô vuông sẽ có kích cỡ:

$$nbox + nbox \times 4 + nclass$$

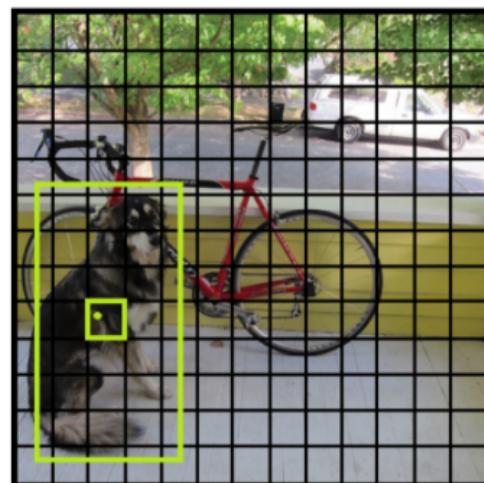
Trong trường hợp $nbox=2$, $nclass=20$. Kết quả đầu ra sẽ có dạng như hình 3.3:

P₁	P₂	x₁	y₁	w₁	h₁	x₂	y₂	w₂	h₂	P(class ₁)	...	P(class ₂₀)
----------------------	----------------------	----------------------	----------------------	----------------------	----------------------	----------------------	----------------------	----------------------	----------------------	------------------------	-----	-------------------------

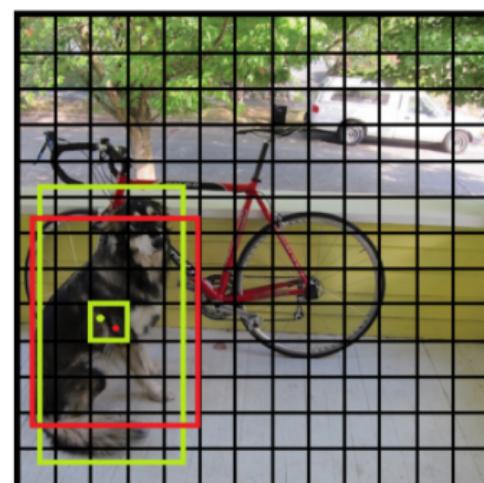
Hình 3.3: Ví dụ về kết quả đầu ra của YOLO khi nbox=2, nclass=20

Trong đó:

- P_k là xác suất có đối tượng trong khu vực thứ k .
- x_k, y_k, w_k, h_k lần lượt là vị trí tương đối hoành độ, tung độ của tâm so với ô lưới và vị trí tương đối của chiều rộng, chiều dài của hộp giới hạn so với ảnh.
- $P(class_k)$ là xác suất khi ô vuông có đối tượng thì xác suất để đối tượng đó thuộc về lớp thứ k là bao nhiêu.



Hình 3.4: Ô vuông trong hệ thống lưới

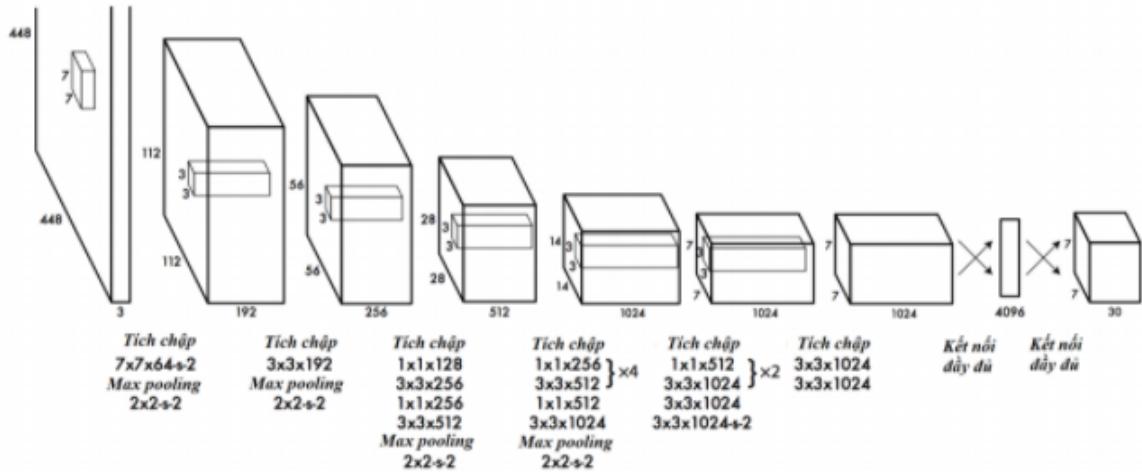


Hình 3.5: Dự đoán 2 hộp giới hạn cho mỗi ô vuông

Tóm lại, với đầu vào là ảnh ta sẽ có kết quả đầu ra là một ma trận hai chiều $M \times N$ với $M = S \times S$ và $N = n_{box} + 4 * n_{box} + class$.

Thiết kế mạng

Kiến trúc mạng được lấy cảm hứng từ mô hình phân loại ảnh GoogleNet. Mạng YOLOv1 sử dụng có 24 lớp tích chập và sau đó là 2 lớp kết nối đầy đủ.



Hình 3.6: Kiến trúc mạng YOLOv1

YOLOv1 không sử dụng hàm kích hoạt ở lớp cuối cùng, còn các lớp khác sử dụng hàm leaky:

$$\phi(x) = \begin{cases} x, & \text{Nếu } x > 0 \\ 0.1x, & \text{Nếu } x \leq 0 \end{cases}$$

Hàm măt mát

Các tác giả sử dụng hàm măt mát là hàm tổng bình phương sai số bởi vì nó dễ dàng tối ưu. Tuy nhiên nó không hoàn toàn phù hợp. Nó cân bằng lỗi định vị đối tượng với lỗi phân loại. Ngoài ra, trong mọi hình ảnh, nhiều ô lướt không chứa bất kỳ đối tượng nào. Điều này làm tăng điểm "độ tin cậy" của các ô đó về 0, thường áp đảo gradient từ các ô có chứa các đối tượng. Có thể dẫn đến sự không ổn định của mô hình. Để khắc phục điều này, YOLO tăng tổn thất từ dự đoán tọa độ hộp giới hạn và giảm măt mát cho các dự đoán hộp không chứa đối tượng. Các tác giả sử dụng hai tham số, $\lambda_{coord} = 5$ và $\lambda_{noobj} = 0.5$ để làm điều đó.

Hàm mất mát của YOLO được định nghĩa như sau [1]:

$$\begin{aligned}
& \lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^B \mathbb{I}_{ij}^{\text{obj}} [(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2] \\
& + \lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^B \mathbb{I}_{ij}^{\text{obj}} \left[(\sqrt{w_i} - \sqrt{\hat{w}_i})^2 + (\sqrt{h_i} - \sqrt{\hat{h}_i})^2 \right] + \sum_{i=0}^{S^2} \sum_{j=0}^B \mathbb{I}_{ij}^{\text{obj}} (C_i - \hat{C}_i)^2 \\
& + \lambda_{\text{noobj}} \sum_{i=0}^{S^2} \sum_{j=0}^B \mathbb{I}_{ij}^{\text{noobj}} (C_i - \hat{C}_i)^2 + \sum_{i=0}^{S^2} \mathbb{I}_i^{\text{obj}} \sum_{c \in \text{classes}} (p_i(c) - \hat{p}_i(c))^2
\end{aligned}$$

trong đó:

- $\mathbb{I}_i^{\text{obj}} = 1$ nếu ô vuông i đang xét chứa đối tượng, và bằng 0 nếu ngược lại.
- $\hat{p}_i(c)$ là xác suất ô vuông i có chứa đối tượng thuộc lớp c trong tập các lớp mà ta đang phân lớp.
- $p_i(c) = 1$ nếu ô vuông i có chứa đối tượng thuộc lớp c , và bằng 0 nếu ngược lại.
- $\mathbb{I}_{ij}^{\text{obj}} = 1$ nếu bộ dự đoán hộp giới hạn thứ j trong ô i "chịu trách nhiệm" cho dự đoán đó.
- $\mathbb{I}_{ij}^{\text{noobj}} = 1 - \mathbb{I}_{ij}^{\text{obj}}$
- $C_i = 1$ nếu ô i chứa đối tượng và bằng 0 nếu ngược lại.
- \hat{C}_i là xác suất mô hình dự đoán ô vuông i chứa đối tượng.

Hạn chế của YOLOv1

YOLOv1 áp đặt các ràng buộc về không gian trên những hộp giới hạn, mỗi ô lưới chỉ có thể dự đoán rất ít hộp giới hạn và duy nhất một lớp (class). Các ràng buộc này hạn chế khả năng nhận biết số đối tượng nằm gần nhau, cũng như đối với các đối tượng có kích thước nhỏ.

YOLOv1 sử dụng các đặc trưng tương đối thô để dự đoán hộp giới hạn, do mô hình sử dụng nhiều lớp downsampling từ ảnh đầu vào. Bởi các hạn chế này của mô hình khi huấn luyện để dự đoán hộp giới hạn từ tập dữ liệu, dẫn đến YOLOv1 không thực sự tốt trong việc nhận diện các đối tượng với tỉ lệ hình khối mới hoặc bất thường so với tập dữ liệu.

Ngoài ra, trong quá trình huấn luyện, hàm mất mát không có sự đánh giá riêng biệt giữa lỗi của hộp giới hạn kích thước nhỏ so với lỗi của hộp giới hạn kích thước lớn. Việc coi chúng như cùng loại và tổng hợp lại làm ảnh hưởng đến độ chính xác toàn cục của mạng. Lỗi nhỏ trên box lớn nhìn chung ít tác hại, nhưng lỗi nhỏ với box rất nhỏ sẽ đặc biệt ảnh hưởng đến giá trị IoU.

3.2 YOLO phiên bản 2

YOLOv2 được đặt tên là **YOLO9000** [2], một kiến trúc phát hiện đối tượng real-time có thể phát hiện hơn 9000 đối tượng khác nhau. Vượt trội hơn các phương pháp như Faster-RCNN hay SSD trong khi thời gian chạy nhanh hơn.

Batch normalization

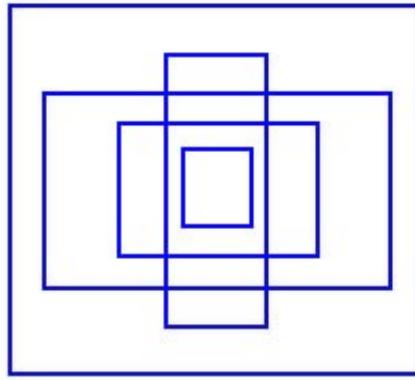
Batch normalization [10] dẫn đến những cải tiến đáng kể trong hội tụ, đồng thời loại bỏ các hình thức regularization khác. Bằng cách thêm batch normalization vào sau tất cả các lớp convolution trong yolo giúp cải thiện mAP lên 2%. Batch normalization cũng giúp regularize mô hình, mạng không cần sử dụng dropout để tăng tính phổ quát.

High resolution classifier

YOLO được huấn luyện với 2 pha. Pha đầu sẽ huấn luyện một mạng phân lớp với ảnh đầu vào kích thước nhỏ (224×224) và pha sau sẽ loại bỏ lớp kết nối đầy đủ và sử dụng mạng phân lớp này như phần khung xương (backbone) để huấn luyện mạng detection. Lưu ý rằng ảnh đầu vào kích thước nhỏ cũng thường được sử dụng để huấn luyện các mạng phân lớp, mà sau đó sẽ được sử dụng như pretrained model cho phần backbone của các mạng detection khác. Ở pha sau YOLO trước hết finetune mạng backbone dưới ảnh đầu vào kích thước lớn hơn là 448×448 , để mạng "quen" dần với kích thước ảnh đầu vào lớn, sau đó mới sử dụng kết quả này để huấn luyện cho quá trình detection. Điều này giúp tăng mAP của YOLOv2 lên khoảng 4%.

Sử dụng Anchor box

Trong YOLOv2, tác giả loại bỏ lớp kết nối đầy đủ ở giữa mạng và sử dụng kiến trúc anchor box để dự đoán các hộp giới hạn. Việc dự đoán các offset so với anchor box sẽ dễ dàng hơn nhiều so với dự đoán toạ độ hộp giới hạn. Thay đổi này làm giảm mAP đi một chút nhưng làm recall tăng lên. Trong bài báo, anchor còn được gọi là prior.



Hình 3.7: YOLOv2 dự đoán cho 5 anchor box tại mỗi ô lưới

Dismention Clusters

Thay vì phải chọn anchor box bằng tay, YOLOv2 sử dụng thuật toán k-means để đưa ra các lựa chọn anchor box tốt nhất cho mạng. Việc này tạo ra mean IoU tốt hơn.

Direct location prediction

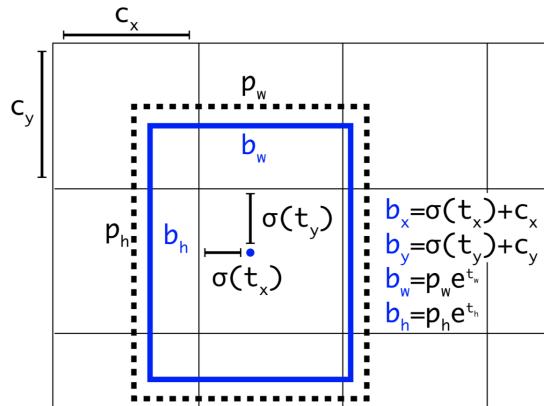
YOLOv1 không có các hạn chế trong việc dự đoán vị trí của hộp giới hạn. Khi các trọng số được khởi tạo ngẫu nhiên, hộp giới hạn có thể được dự đoán ở bất kỳ đâu trong ảnh. Điều này khiến mô hình không ổn định trong giai đoạn đầu của quá trình huấn luyện. Vị trí của hộp giới hạn có thể ở rất xa so với vị trí của ô lưới. Trong mạng Region proposal, mạng dự đoán các giá trị t_x và t_y và toạ độ tâm (x, y) bằng cách:

$$\begin{aligned} x &= (t_x * w_a) - x_a \\ y &= (t_y * h_a) - y_a \end{aligned}$$

YOLOv2 dự đoán 5 hộp giới hạn trên mỗi ô lưới của feature map, sử dụng hàm sigmoid (σ) để hạn chế giá trị trong khoảng 0 đến 1, từ đó có thể hạn chế các dự đoán hộp giới hạn ở xung quanh ô lưới, từ đó giúp mô hình ổn định hơn trong quá trình huấn luyện.

Cho anchor box có kích thước (p_w, p_h) nằm tại ô lưới với vị trí top left là (c_x, c_y) , mô hình sẽ dự đoán các offset và scale t_x, t_y, t_w, t_h và hộp giới hạn (b_x, b_y, b_w, b_h) . Độ tự tin của dự đoán là $\sigma(t_o)$.

$$\begin{aligned} b_x &= \sigma(t_x) + c_x \\ b_y &= \sigma(t_y) + c_y \\ b_w &= p_w e^{t_w} \\ b_h &= p_h e^{t_h} \\ \text{Pr(Object).IoU(b, object)} &= \sigma(t_o) \end{aligned}$$



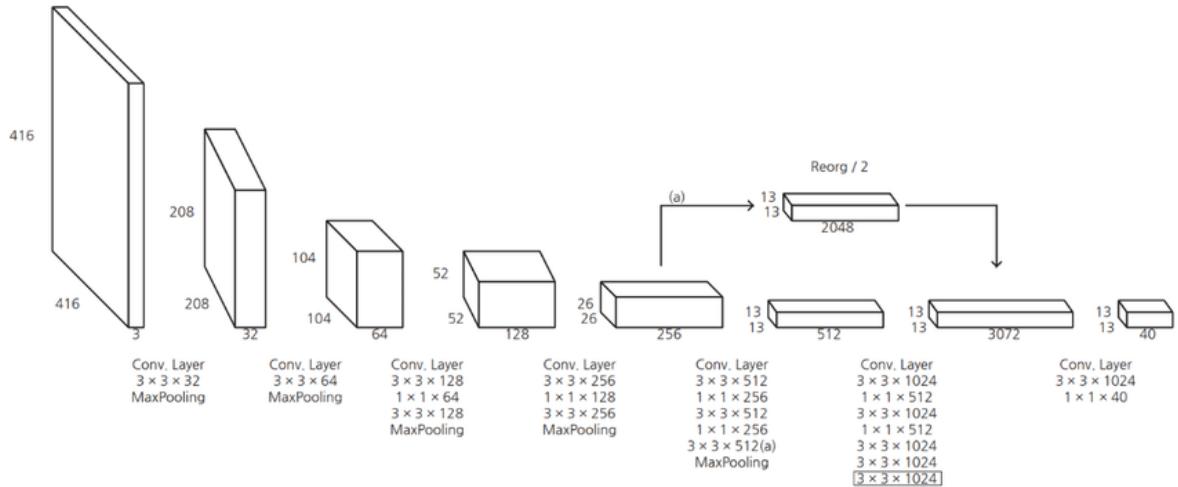
Hình 3.8: Dự đoán hộp giới hạn trong YOLOv2

YOLOv2 đã có thêm 5% mAP khi áp dụng phương pháp này.

Fine-Grained Features

YOLOv2 sử dụng feature map 13×13 để đưa ra các dự đoán, lớn hơn 7×7 của YOLOv1.

Faster R-CNN và SSD đưa ra dự đoán ở nhiều tầng khác nhau trong mạng để tận dụng các feature map ở các kích thước khác nhau. Yolov2 cũng kết hợp các feature ở các tầng khác nhau lại để đưa ra dự đoán, cụ thể kiến trúc nguyên bản của Yolov2 kết hợp feature map 26×26 lấy từ đoạn gần cuối với feature map 13×13 ở cuối để đưa ra các dự đoán. Cụ thể là các feature map này sẽ được ghép vào nhau (concatenate) để tạo thành một khối sử dụng cho dự đoán.



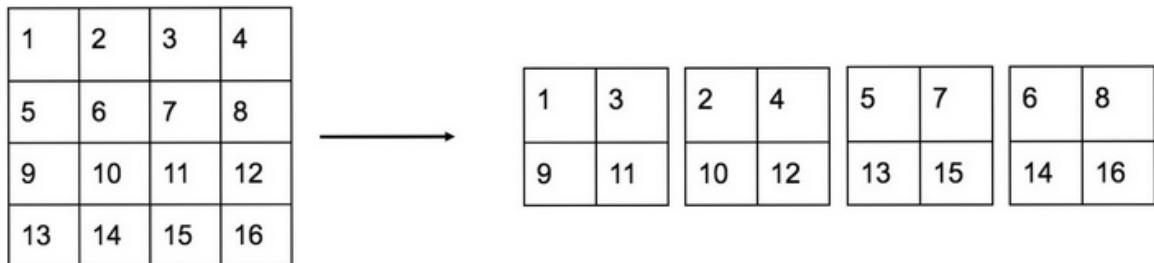
Hình 3.9: Kiến trúc Yolov2

Vậy làm thế nào để concatenate được 2 feature map kích thước $26 \times 26 \times m$ và $13 \times 13 \times n$ để trở thành một feature map $13 \times 13 \times p$?

Thông thường việc concatenate 2 feature map chỉ thực hiện được khi chúng có cùng

chiều rộng và chiều dài. Tuy nhiên, trong Yolov2 tác giả sử dụng lớp **Reorg**, tuy nhiên lại không mô tả kỹ về kĩ thuật này trong paper. Thực ra **Reorg** chỉ là kĩ thuật tổ chức lại bộ nhớ để biến feature map 26×26 thành 13×13 với chiều sâu lớn hơn để có thể thực hiện phép concatenate với feature map 13×13 ở cuối.

Trong trường hợp tổng quát của phép **Reorg**, ta sẽ biến feature map kích thước $[N, C, H, W]$ thành kích thước $[N, C \times s^2, \frac{H}{s}, \frac{W}{s}]$, tức là số lượng tham số trong feature map vẫn được giữ nguyên. Khi ta muốn giảm kích thước dài, rộng đi mỗi cạnh 2 lần thì số channel phải được tăng lên 4 lần. Việc biến đổi này hoàn toàn không giống phép resize trong xử lý ảnh. Ví dụ:



Hình 3.10: Kĩ thuật Reorg trong Yolov2

Đây là một lát cắt (channel) của feature map kích thước 4×4 . Để đưa về kích thước 2×2 , tức là giảm chiều rộng đi 2 lần và chiều dài đi 2 lần, ta tách channel của feature map 4×4 thành 4 ma trận như hình vẽ trên, ứng với 4 channel chiều sâu của feature map 2×2 mới. Vị trí các giá trị trong mỗi channel của feature map 2×2 mới sẽ lấy thừa thớt trên feature map 4×4 ban đầu với stride = 2 theo 2 trục dài và rộng. Bằng kĩ thuật "Add fine-grained features", performance của mạng YOLOv2 được tăng thêm 1%.

Multi-Scale Training

Sau khi thêm kĩ thuật anchor box cho YOLOv2, tác giả đã thay đổi đầu vào của mạng thành 416×416 thay vì 448×448 . Tuy vậy, YOLOv2 được thiết kế chỉ gồm các lớp tích chập và pooling nên có thể thích ứng với nhiều kích thước ảnh đầu vào khác nhau. Tác giả đã huấn luyện mạng trên nhiều kích thước ảnh khác nhau để tăng khả năng thích ứng của YOLOv2 với đa dạng kích thước ảnh.

Light-weight backbone

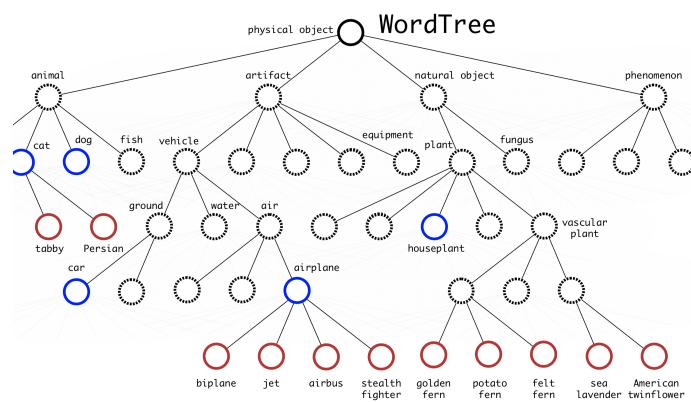
Điểm cải tiến của YOLOv2 còn phải kể đến backbone mới có tên Darknet-19. Mạng này bao gồm 19 lớp convolution và 5 lớp max pooling tạo ra tốc độ nhanh hơn phiên bản YOLO trước.

Type	Filters	Size/Stride	Output
Convolutional	32	3×3	224×224
Maxpool		$2 \times 2/2$	112×112
Convolutional	64	3×3	112×112
Maxpool		$2 \times 2/2$	56×56
Convolutional	128	3×3	56×56
Convolutional	64	1×1	56×56
Convolutional	128	3×3	56×56
Maxpool		$2 \times 2/2$	28×28
Convolutional	256	3×3	28×28
Convolutional	128	1×1	28×28
Convolutional	256	3×3	28×28
Maxpool		$2 \times 2/2$	14×14
Convolutional	512	3×3	14×14
Convolutional	256	1×1	14×14
Convolutional	512	3×3	14×14
Convolutional	256	1×1	14×14
Convolutional	512	3×3	14×14
Maxpool		$2 \times 2/2$	7×7
Convolutional	1024	3×3	7×7
Convolutional	512	1×1	7×7
Convolutional	1024	3×3	7×7
Convolutional	512	1×1	7×7
Convolutional	1024	3×3	7×7
Convolutional	1000	1×1	7×7
Avgpool		Global	1000
Softmax			

Hình 3.11: Kiến trúc Darknet-19

WordTree

YOLO9000 đưa ra cách kết hợp các dataset khác với ImageNet để có thể phát hiện nhiều lớp hơn. Tác giả tạo một đồ thị có hướng gọi là WordTree như hình 3.12. Để có thể trộn được các nhãn từ tập ImageNet (1000 lớp) với COCO/PASCAL (100 lớp), tác giả dựa vào WordNet để xây dựng quan hệ giữa các lớp, từ đó có thể huấn luyện mạng nhận dạng các class có quan hệ với nhau.



Hình 3.12: WordTree - YOLO9000

Ví dụ để dự đoán xác suất rơi vào một lớp, ta nhân các xác suất từ nhánh gốc của đồ thị và dừng lại khi xác suất vào các class nhánh nhỏ hơn một ngưỡng nào đó.

3.3 YOLO phiên bản 3

YOLOv3 có kiến trúc khá giống YOLOv2. Tác giả đã thêm các cải tiến mới trong các nghiên cứu gần đây vào YOLOv2 để tạo ra YOLOv3. Các cải tiến đó bao gồm:

Thay softmax bằng các logistic classifier rời rạc

YOLOv3 sử dụng các logistic classifier thay vì softmax cho việc classify đối tượng. Việc này cho hiệu quả tốt hơn nếu các nhãn không loại trừ lẫn nhau, tức là có thể có đối tượng cùng thuộc 2 hay nhiều lớp khác nhau. Ví dụ: nhãn đầu ra có thể là "người đi bộ" và "trẻ em" không phải là nhãn không độc quyền. (tổng của đầu ra hiện có thể lớn hơn 1.) YOLOv3 thay thế hàm softmax bằng các bộ phân loại logistic độc lập để tính toán khả năng đầu vào thuộc về một nhãn cụ thể. Thay vì sử dụng sai số bình phương trung bình để tính toán sai số phân loại, YOLOv3 sử dụng hàm mất mát là cross entropy cho mỗi nhãn:

$$\text{CrossEntropy} = - \sum y * \log(y_{\text{predict}})$$

Hàm cross entropy được sử dụng nhờ độ dốc của nó lớn hơn độ dốc của hàm mất mát MSE, nhờ đó đạo hàm của hàm cross entropy cũng lớn hơn tạo điều kiện cập nhật tham số mô hình tốt hơn.

Backbone mới - Darknet-53

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

Hình 3.13: Kiến trúc Darnknet-53

Backbone được thiết kế lại với việc thêm các residual blocks (kiến trúc sử dụng trong ResNet).

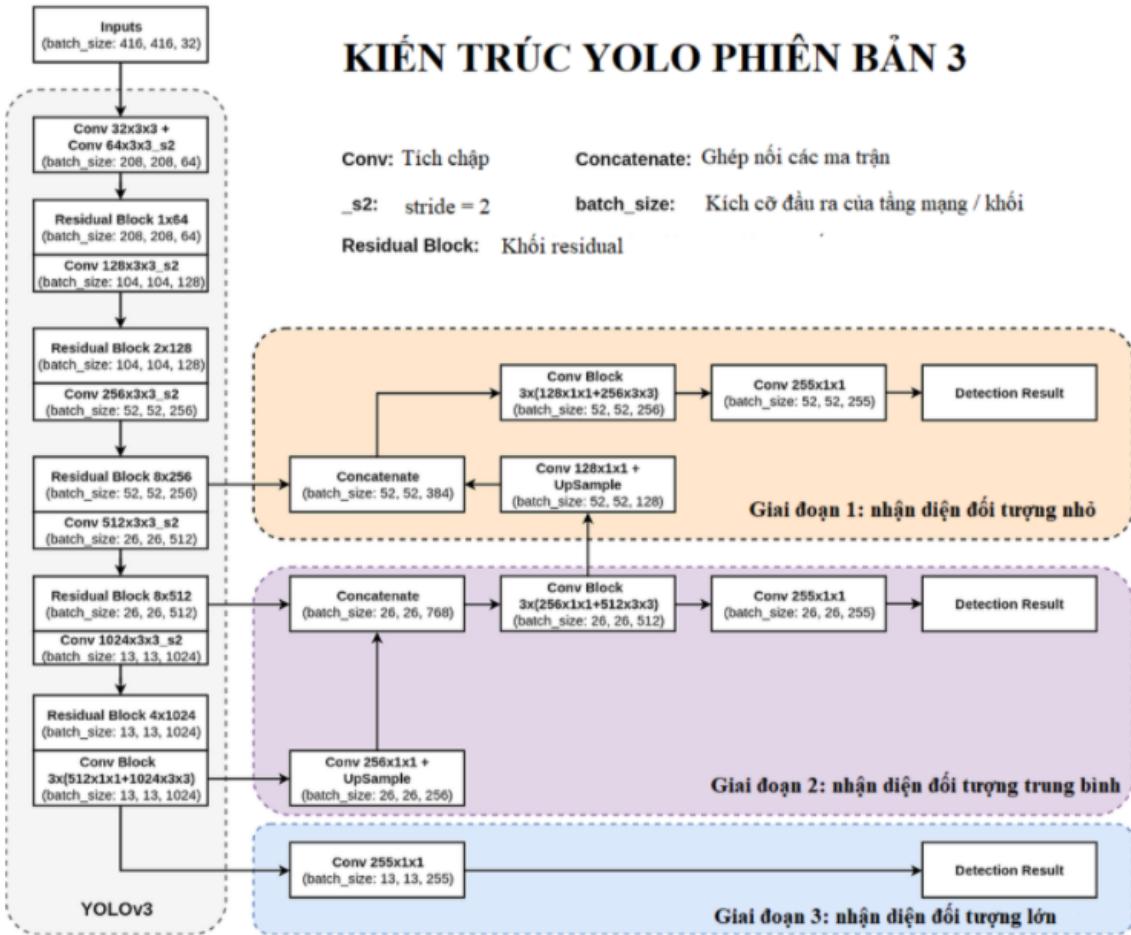
Khối residual là khối các tầng tích chập với ý tưởng chính là kết nối tầng đầu của khối và tầng cuối của khối bằng phép cộng ma trận. Giả sử tầng đầu vào của khối residual là x và tầng cuối của khối là $F(x)$. Khi đó ta sẽ lấy $H(x) = F(x) + x$ để làm tầng đầu vào cho lớp kế tiếp.

Khối residual ra đời trong bối cảnh các mô hình học sâu đang có bước chững lại. Nguyên nhân là do khi ta thêm tầng mạng cho mô hình học sâu sẽ dẫn đến việc mất mát thông tin của đối tượng. Việc xuất hiện khối residual làm cho số tầng mạng tăng lên đáng kể mà tránh mất mát thông tin. Từ đó, tạo bước đệm vững chãi cho phép mô hình học sâu tiếp tục có không gian phát triển.

Skip-layer concatenation

YOLOv3 cũng thêm các liên kết giữa các lớp dự đoán. Mô hình upsample các lớp dự đoán ở các tầng sau và sau đó concatenate với các lớp dự đoán ở các tầng trước đó. Phương pháp này giúp tăng độ chính xác khi dự đoán các đối tượng nhỏ.

KIẾN TRÚC YOLO PHIÊN BẢN 3



Hình 3.14: Kiến trúc YOLOv3

3.4 YOLO phiên bản 4

YOLOv4 đưa ra mô hình phát hiện đối tượng thời gian thực trên GPU thông thường, và sử dụng 1 GPU cho huấn luyện. YOLOv4 cải thiện cả về tốc độ lẫn độ chính xác so với YOLOv3. Cải tiến của YOLOv4 như sau:

1. YOLOv4 phát triển mô hình phát hiện đối tượng hiệu quả và mạnh mẽ. Giúp mọi người có thể sử dụng GPU 1800 Ti hoặc 2080 Ti để huấn luyện mô hình phát hiện đối tượng nhanh và chính xác.
2. YOLOv4 sử dụng các phương pháp **Bag of Freebies** và **Bag of Specials** trong quá trình huấn luyện.
 - **Bag of Freebies:** Những phương pháp giúp cải thiện kết quả suy diễn mà không làm ảnh hưởng tới tốc độ. Những phương pháp này thường là data augmentation, class imbalance, cost function, soft labeling, ...
 - **Bag of Specials:** Những phương pháp hi sinh một chút tốc độ suy diễn mà làm cải thiện độ chính xác của mô hình đáng kể. Những phương pháp này bao gồm tăng receptive field, sử dụng attention, feature intergration (kết hợp thông tin của các feature maps với nhau) như skip-connection & FPN (Feature Paramyd Network), hậu xử lý như NMS (Non Maximum Suppression).
3. YOLOv4 sửa đổi các kiến trúc hiện đại và làm cho chúng đơn giản hơn và phù hợp với đào tạo GPU đơn, bao gồm CBN, PAN, SAM, ...

Phân loại các mô hình phát hiện đối tượng

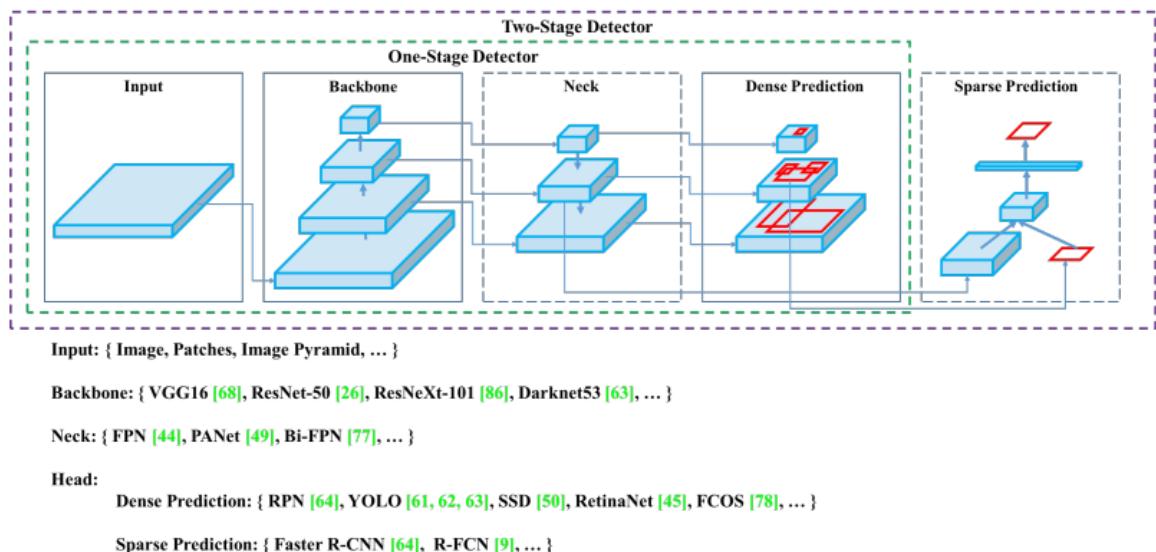
Một mô hình phát hiện đối tượng thường chia làm hai phần, một *backone* sử dụng pre-trained trên ImageNet và một *head* sử dụng để dự đoán lớp và hộp giới của đối tượng. Với một số trình phát hiện đối tượng chạy trên nền tảng GPU, backbone có thể là VGG, ResNet, ResNeXt hoặc DenseNet. Với một số trình phát hiện đối tượng chạy trên nền tảng CPU, backbone có thể là SqueezeNet, MobileNet, hoặc ShuffleNet. Về phần *head*, thường được phân làm 2 loại:

- **Bộ phát hiện đối tượng hai giai đoạn:** Diễn hình là các thuật toán R-CNN, Fast R-CNN, Faster R-CNN, R-FCN, và Libra R-CNN. Gọi là hai giai đoạn do cách mà mô hình xử lý để lấy được ra các vùng có khả năng chứa vật thể từ ảnh. Ví dụ với Faster R-CNN thì trong giai đoạn 1, ảnh sẽ được đưa qua một mạng gọi là region proposal network (RPN) với mục tiêu trích xuất ra các vùng chứa đối tượng dựa vào anchor. Sau khi thu được các vùng đặc trưng từ RPN, mô hình Faster R-CNN sẽ thực hiện tiếp việc phân loại đối tượng và định vị đối tượng nhờ vào việc chia làm 2 nhánh tại phần cuối của mô hình.

- **Bộ phát hiện đối tượng một giai đoạn:** Diển hình là YOLO, SSD và RetinaNet. Việc thiết kế mô hình hoàn toàn không có phần trích chọn đặc trưng của các vùng có khả năng chứa đối tượng. Các mô hình 1 giai đoạn coi việc định vị đối tượng như một bài toán hồi quy. Các mô hình dạng này thường nhanh hơn nhưng độ chính xác kém hơn so với các mô hình 2 giai đoạn.

Hình 3.15 mô tả các thành phần cấu tạo nên các mô hình phát hiện đối tượng. Các trình phát hiện đối tượng được phát triển trong những năm gần đây thường chèn một số tầng giữa backbone và head, và các tầng này thường sử dụng để collect feature map từ các giai đoạn khác nhau. Chúng ta gọi nó là **neck** của trình phát hiện đối tượng. Thông thường, một neck bao gồm một số đường bottom-up và một số đường top-down. Các mạng trang bị cơ chế này bao gồm Feature Pyramid Network (FPN), Path Aggregation Network (PAN), BiFPN, và NAS-FPN.

Ngoài những mô hình bên trên, một số nhà nghiên cứu tập trung và xây dựng một backbone mới (DetNet, DetNAS) hoặc một mô hình hoàn toàn mới (SpineNet, HitDetector) cho phát hiện đối tượng.



Hình 3.15: Kiến trúc chung cho các trình phát hiện đối tượng

Tóm lại, một trình phát hiện đối tượng bao gồm một số phần:

- **Input:** Image, Patches, Image Pyramid
- **Backbones:** VGG16, ResNet-50, SpineNet, EfficientNET-N0/B7, CSPResNeXt50, CSPDarknet53
- **Neck:**
 - **Additional blocks:** SPP, ASPP, RFB, SAM

- **Path-aggregation blocks:** FPN, PAN, NAS-FPN, Fully connected FPN, BiFPN, ASFF, SFAM
- **Heads:**
 - **Dense Prediction (one-state):**
 - * RPN, SSD, YOLO, RetinaNet (anchor based)
 - * CornerNet, CenterNet, MatrixNet, FCOS (anchor free)
 - **Sparse Prediction (two-state):**
 - * Faster R-CNN, R-FCN, Mask R-CNN (anchor based)
 - * RepPoints (anchor free)

Các kỹ thuật sử dụng trong YOLOv4

YOLOv4 sử dụng các mô hình, kỹ thuật như sau:

- Backbone: CSPDarknet53
- Neck: SPP, PAN
- Head: YOLOv3
- Bag of Freebies (BoF) cho backbone: CutMix và Mosaic data augmentation, DropBlock regularization, Class Label smoothing.
- Bag of Specials (BoS) cho backbone: Hàm kích hoạt Mish, Cross-state partial connections (CSP), Multi input weighted residual connections (MiWRC)
- Bag of Freebies (BoF) cho detector: CIoU-loss, CmBN, DropBlock regularization, Mosaic data augmentation, Self-Adversarial Training, Eliminate grid sensitivity, sử dụng multiple anchors cho một hộp giới hạn, Cosine annealing scheduler, tối ưu tham số sử dụng giải thuật di truyền, huấn luyện với các hình ảnh đầu vào ngẫu nhiên.
- Bag of Specials (BoS) for detector: Hàm kích hoạt Mish, SPP-block, SAM-block, PAN path-aggregation block, DIoU-NMS

Tại thời điểm viết báo cáo này, YOLOv5 đang được phát triển và kỳ vọng sẽ đạt hiệu quả hơn YOLOv4. Tuy nhiên, chưa có một bài báo chính thức nào trình bày về các kỹ thuật cải tiến ở YOLOv5. Vì vậy, báo cáo này tạm dừng trình bày kỹ thuật ở phiên bản thứ 4 của YOLO.

Chương 4

Tổng quan về học tích cực

Sự ra đời của học máy là một bước tiến thay đổi nền khoa học công nghệ của nhân loại. Để tiếp tục phát triển hơn nữa, chúng ta cần phải tiếp cận những phương pháp có thể tối ưu được thời gian, tốc độ huấn luyện và dự đoán, đồng thời cải thiện được độ chính xác của mô hình. Một trong những kỹ thuật đang được nghiên cứu khai thác để tinh chỉnh cho hiệu suất của các mô hình đó là học tích cực.

Nội dung chương 4 sẽ tìm hiểu về khái niệm, phương thức hoạt động của học tích cực, ...

4.1 Giới thiệu về học tích cực

4.1.1 Khái niệm học tích cực

Học tích cực (còn được gọi là “học truy vấn”, hoặc đôi khi là “thiết kế thử nghiệm tối ưu” trong tài liệu thống kê) là một trường con của học máy nói riêng và trí tuệ nhân tạo nói chung. Giả thuyết quan trọng là, nếu thuật toán học được phép chọn dữ liệu mà nó học được thì nó sẽ hoạt động tốt hơn với ít đào tạo hơn. Tại sao đây là thuộc tính mong muốn để học thuật toán có? Hãy xem xét rằng, để bất kỳ hệ thống học tập có giám sát nào hoạt động tốt, nó thường phải được đào tạo trên hàng trăm (thậm chí hàng nghìn) phiên bản được gắn nhãn. Đôi khi những nhãn này có ít, chẳng hạn như cờ “spam” mà chúng ta đánh dấu trên các email không mong muốn hoặc xếp hạng năm sao mà bạn có thể cho các bộ phim trên trang web mạng xã hội. Hệ thống học tập sử dụng các cờ và xếp hạng này để lọc email rác của chúng ta tốt hơn và đề xuất các bộ phim chúng ta có thể thích. Trong những trường hợp này, chúng ta cung cấp các nhãn như vậy miễn phí, nhưng đối với nhiều nhiệm vụ học tập có giám sát phức tạp hơn, các phiên bản có nhãn là rất khó, tốn thời gian hoặc tốn kém để có được. Đây là vài ví dụ:

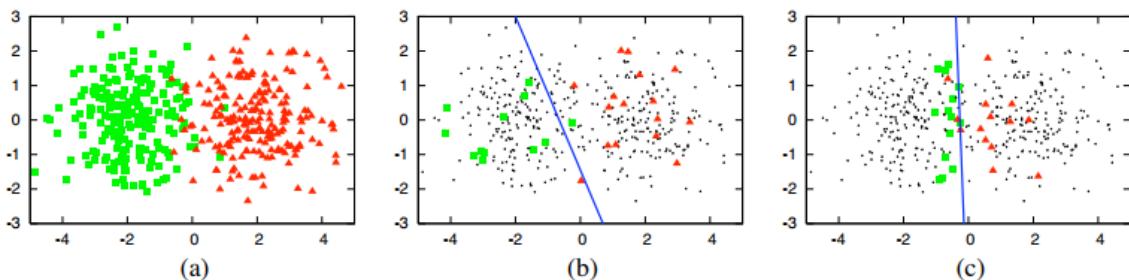
- **Nhận dạng giọng nói:** Việc gán nhãn chính xác các cách phát âm là cực kỳ mất thời gian và đòi hỏi các nhà ngôn ngữ học được đào tạo. Việc chú thích ở cấp độ từ có thể mất nhiều thời gian hơn 10 lần so với âm thanh thực tế (ví dụ:

một phút bài phát biểu mất mười phút để gắn nhãn) và chú thích âm vị có thể lâu gấp 400 lần (ví dụ: gần bảy giờ) . Vấn đề là phức tạp đối với các ngôn ngữ hoặc phương ngữ hiếm.

- **Trích xuất thông tin:** Hệ thống khai thác thông tin tốt phải được đào tạo bằng cách sử dụng các tài liệu có nhãn với các chú thích chi tiết. Người dùng đánh dấu các thực thể hoặc mối quan hệ quan tâm trong văn bản, chẳng hạn như tên người và tổ chức, hoặc liệu một người có làm việc cho một tổ chức cụ thể hay không. Việc xác định các thực thể và các mối quan hệ có thể mất nhiều thời gian. Chú thích cho các lĩnh vực kiến thức khác có thể yêu cầu chuyên môn bổ sung, ví dụ: chú thích đề cập đến gen và bệnh để khai thác thông tin y sinh thường yêu cầu trình độ chuyên môn cao.
- **Phân loại và lọc:** Học cách phân loại tài liệu (ví dụ: bài báo hoặc trang web) hoặc bất kỳ loại phương tiện nào khác (ví dụ: tệp hình ảnh, âm thanh và video) yêu cầu người dùng gắn nhãn mỗi tài liệu hoặc tệp phương tiện bằng các nhãn cụ thể, như “có liên quan” hoặc “không liên quan”. Việc phải chú thích hàng ngàn trường hợp này có thể tẻ nhạt và thậm chí là thừa.

Các hệ thống học tập tích cực cố gắng vượt qua các bước đầu tiên về vấn đề gắn nhãn bằng cách yêu cầu các truy vấn để được gắn nhãn bởi một người bất kỳ. Bằng cách này, học tích cực hướng tới mục tiêu đạt được độ chính xác cao bằng cách sử dụng càng ít mẫu được gắn nhãn càng tốt, do đó giảm thiểu chi phí lấy dữ liệu được gắn nhãn. Học tập tích cực được thúc đẩy tốt trong nhiều vấn đề học máy hiện đại, nơi dữ liệu có thể dồi dào nhưng nhãn thì khan hiếm hoặc tốn kém để lấy. Lưu ý rằng loại hình học tập tích cực này không có liên quan về mặt tinh thần, chúng ta không được nhầm lẫn, với họ các kỹ thuật hướng dẫn có cùng tên trong các tài liệu giáo dục.

4.1.2 Ví dụ về học tích cực



Hình 4.1: Ví dụ về học tích cực

Chúng ta có thể thấy ở hình 4.1 (a) có 2 cụm dữ liệu màu xanh và màu đỏ. Mục tiêu là cần tìm đường ranh giới tuyến tính để phân lớp 2 bộ dữ liệu này nhưng không thể

gán nhãn toàn bộ dữ liệu vì chi phí tốn kém mà chỉ có thể gán nhãn cho một tập con của bộ dữ liệu. Sau đó sử dụng phần dữ liệu đã gán nhãn này để phân loại.

Hình 4.1 (b) mô tả đường phân lớp tạo được khi lấy mẫu ngẫu nhiên (Lấy ngẫu nhiên các điểm xanh và đỏ để phân lớp). Hình 4.1 (c) mô tả đường phân lớp tạo được khi lấy mẫu bằng phương pháp học tích cực.

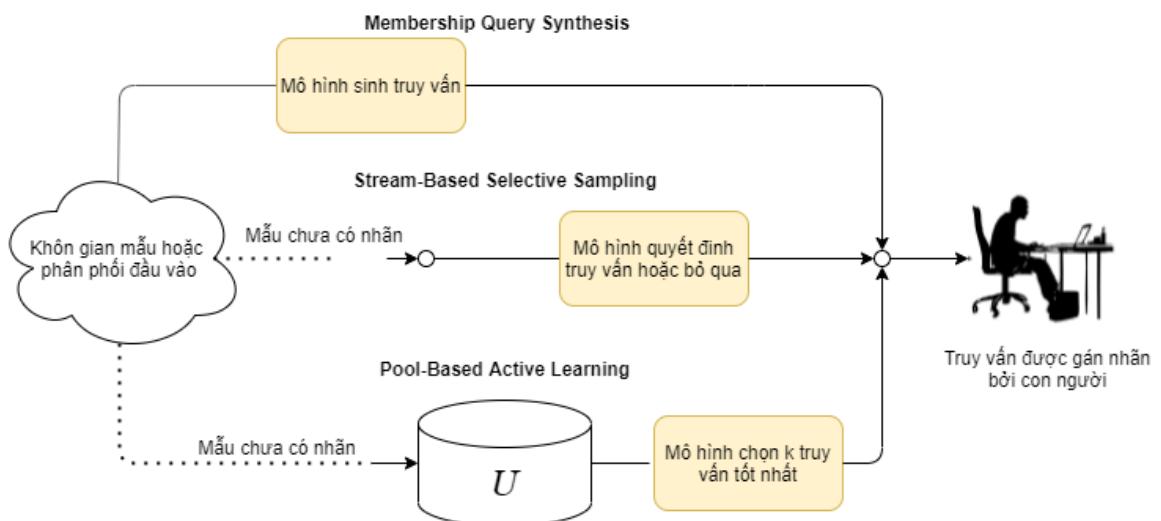
Có thể thấy, rõ ràng cùng với một lượng dữ liệu ít lấy mẫu như nhau, học tích cực cho kết quả tốt hơn so với lấy ngẫu nhiên. Việc áp dụng học tích cực vào các bài toán giúp chúng ta sử dụng ít dữ liệu nhưng vẫn đem lại hiệu quả cao.

4.2 Kịch bản

Có một số tình huống vấn đề khác nhau trong đó người học có thể đặt câu hỏi. Ba kịch bản chính đã được đề cập trong báo cáo là:

- (i) Membership Query Synthesis
- (ii) Stream-Based Selective Sampling
- (iii) Pool-Based Sampling

Hình 4.2 minh họa sự khác biệt giữa ba kịch bản này. Phần còn lại của phần này cung cấp tổng quan về các cài đặt học tập tích cực khác nhau.



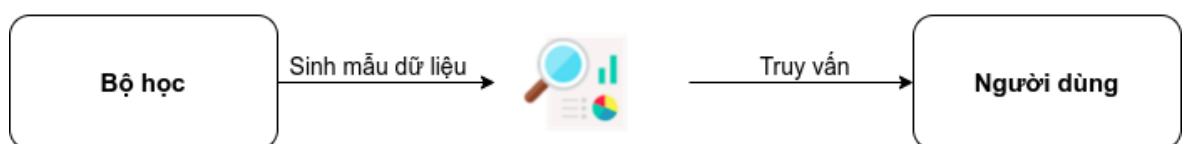
Hình 4.2: Sơ đồ minh họa ba kịch bản học tích cực chính

4.2.1 Membership query synthesis

Một trong những tình huống học tập tích cực đầu tiên được nghiên cứu là học với *membership queries*. Trong cài đặt này, người sử dụng có thể yêu cầu gán nhãn bất kỳ

mẫu nào chưa có nhãn trong không gian các mẫu đầu vào, bao gồm các mẫu mà bộ học tự tạo ra và các mẫu này được xây dựng từ một phân phối tự nhiên cơ bản.

Membership query synthesis là hợp lý đối với nhiều bài toán, tuy nhiên việc gắn nhãn các mẫu tùy ý như vậy có thể gây khó khăn đối với con người. Ví dụ, Baum và Lang (1992) [23] đã sử dụng *membership query synthesis* tương tác với con người để huấn luyện một mạng neural phân loại các ký tự viết tay. Họ đã gặp phải một vấn đề không mong muốn: nhiều hình ảnh truy vấn do bộ học tạo ra không chứa ký hiệu dễ nhận biết, chỉ có các ký tự lai tạo không có ý nghĩa ngữ nghĩa tự nhiên. Tương tự, người ta có thể hình dung được rằng *membership query synthesis* cho các tác vụ xử lý ngôn ngữ tự nhiên có thể tạo ra các luồng văn bản hoặc giọng nói vô nghĩa. Các kịch bản Stream-based selective sampling và Pool-based sampling đã được đề xuất để giải quyết những hạn chế này.

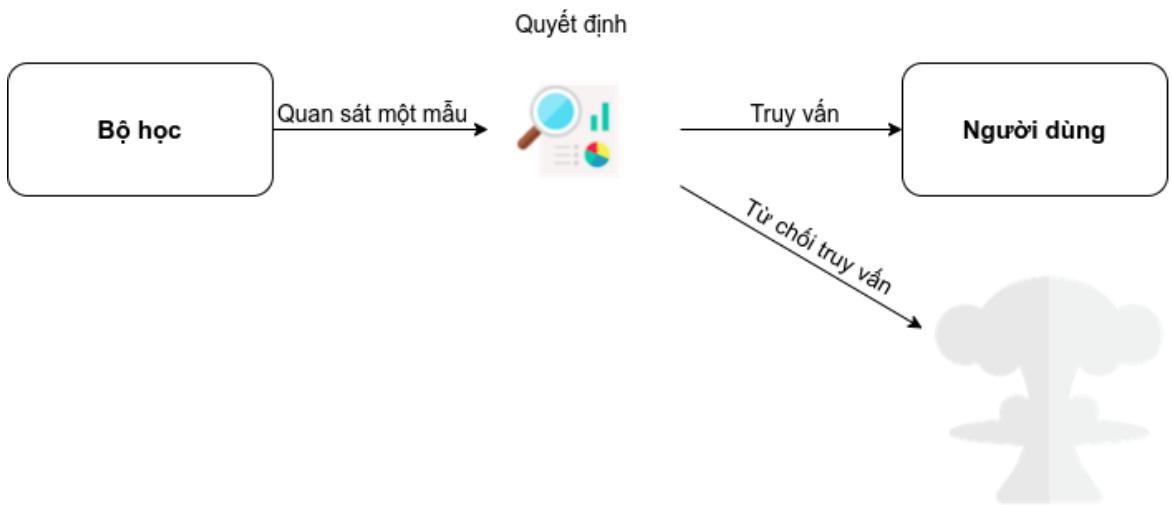


Hình 4.3: Sơ đồ membership query synthesis

4.2.2 Stream-based selective sampling

Một giải pháp thay thế cho việc *membership query synthesis* là *stream-based selective sampling* - lấy mẫu chọn lọc dựa trên luồng. Bộ học lấy mẫu tại một thời điểm nào đó từ các dữ liệu phân tán và cố gắng đưa ra một quyết định trên mẫu này dựa trên lượng thông tin của nó (lượng thông tin này được đánh giá dựa trên các chiến lược truy vấn được giới thiệu trong phần 4.3) và đem nó ra để hỏi người dùng sẽ ghi nhãn cho nó hoặc bỏ qua nó.

Kịch bản *stream-based sampling* đã được nghiên cứu trong một số vấn đề trên thực tế, bao gồm gắn thẻ từng phần của giọng nói, lập lịch cảm biến và học các chức năng xếp hạng để truy xuất thông tin. Fujii và cộng sự. (1998) [24] sử dụng phương pháp lấy mẫu chọn lọc dựa trên luồng để học tích cực xác định nghĩa của từ. Cách tiếp cận này không chỉ làm giảm nỗ lực gán nhãn, mà còn giới hạn kích thước của dữ liệu được sử dụng trong việc áp dụng mô hình láng giềng gần nhất, do đó giải quyết thuật toán phân loại.



Hình 4.4: Sơ đồ stream-base selective sampling

4.2.3 Pool-based sampling

Đối với nhiều vấn đề học trong thực tế, phần lớn dữ liệu trong các bộ dữ liệu không được gắn nhãn có thể được thu thập cùng một lúc. Điều này thúc đẩy học tập tích cực dựa trên nhóm - *pool-based sampling*, giả định rằng có một tập hợp nhỏ dữ liệu có nhãn L và một nhóm lớn dữ liệu không gắn nhãn U có sẵn. Các truy vấn được chọn lọc từ nhóm, thường được giả định là đóng (tức là tĩnh hoặc không thay đổi), mặc dù điều này không hoàn toàn cần thiết. Thông thường, các trường hợp được truy vấn theo kiểu tham lam, theo một thước đo tính thông tin được sử dụng để đánh giá tất cả các trường hợp trong nhóm (hoặc, có thể nếu U rất lớn thì sử dụng một số mẫu con của chúng).

Kịch bản *Pool-based sampling* được nghiên cứu trong các vấn đề thực tế sử dụng học máy như phân loại văn bản, trích rút thông tin, trích rút và phân loại hình ảnh, trích rút và phân loại video, xử lý âm thanh,...

Sự khác biệt chính giữa học tập tích cực dựa trên luồng - *stream-based sampling* và dựa trên nhóm - *pool-based sampling* là ở chỗ *stream-based sampling* quét qua dữ liệu một cách tuần tự và đưa ra quyết định truy vấn riêng lẻ, còn *pool-based sampling* đánh giá và xếp hạng toàn bộ tập hợp trước khi chọn truy vấn tốt nhất. Mặc dù kịch bản dựa trên nhóm thường như phổ biến hơn nhiều trong số các bài báo ứng dụng, người ta cũng có thể thấy các bài đặt mà cách tiếp cận dựa trên luồng là thích hợp hơn với một số trường hợp. Ví dụ, khi bộ nhớ hoặc khả năng xử lý có thể bị hạn chế, như với thiết bị di động và thiết bị nhúng.



Hình 4.5: Sơ đồ pool-base sampling

4.3 Các chiến lược truy vấn trong học tích cực

Các mẫu được chọn phải có rất nhiều thông tin có hiệu quả cho bài toán. Để làm được việc lựa chọn mẫu, có một số phương pháp truy vấn độc lập với kịch bản học tích cực đã giới thiệu ở trên. Đã có nhiều cách đề xuất để xây dựng các chiến lược truy vấn như vậy. Phần này cung cấp tổng quan về các khung truy vấn chung được sử dụng. Chúng ta sử dụng ký hiệu x_A^* để chỉ những truy vấn tốt nhất theo một thuật toán lựa chọn A .

4.3.1 Lấy mẫu không chắc chắn

Có lẽ khung truy vấn đơn giản nhất và được sử dụng phổ biến nhất là *Lấy mẫu không chắc chắn* (*uncertainty sampling*). Trong khung truy vấn này, người dùng truy vấn các trường hợp mà nó ít chắc chắn nhất. Cách tiếp cận này thường đơn giản cho các mô hình học tập theo xác suất. Ví dụ, khi sử dụng một mô hình xác suất cho bài toán phân lớp nhị phân, lấy mẫu không chắc chắn đơn giản là truy vấn các trường hợp mà xác suất của positive gần 0.5 nhất.

Với bài toán có 3 hoặc nhiều nhãn lớp, một biến thể lấy mẫu không chắc chắn là dự đoán xác suất mà kém tin cậy nhất - **least confident**:

$$x_{LC}^* = \operatorname{argmax}_x (1 - P_\theta(\hat{y}|x))$$

Với $\hat{y} = \operatorname{argmax}_y P_\theta(y|x)$, là lớp được dự đoán với xác suất cao nhất trong mô hình θ .

Tuy nhiên, tiêu chí cho chiến lược *least confident* chỉ xem xét các thông tin về lớp có thể xảy ra nhất. Do đó, nó bỏ qua các thông tin về phân phối các lớp còn lại. Để khắc phục điều này, một số nhà nghiên cứu sử dụng biến thể lấy mẫu không chắc chắn multi-class gọi là **margin sampling**:

$$x_M^* = \operatorname{argmin}_x (P_\theta(\hat{y}_1|x) - P_\theta(\hat{y}_2|x))$$

Với \hat{y}_1 và \hat{y}_2 tương ứng là nhãn lớp có khả năng xảy ra cao nhất và thứ hai trong mô hình. *Margin sampling* nhằm mục đích sửa chữa thiếu sót trong chiến lược *least confident*, bằng cách kết hợp với lớp có nhiều khả năng nhất thứ hai. Bằng trực quan, các trường hợp có margin lớn rất dễ dàng, vì trình phân loại có ít ngờ ngờ trong việc

phân biệt 2 nhãn lớp có khả năng xảy ra nhất. Margin nhỏ thì không rõ ràng hơn, do đó biết nhãn đúng sẽ giúp mô hình phân biệt rõ ràng hơn giữa chúng. Tuy nhiên, với bài toán tập nhãn rất lớn, cách tiếp cận margin vẫn bỏ qua phần lớn phân phối cho các lớp còn lại.

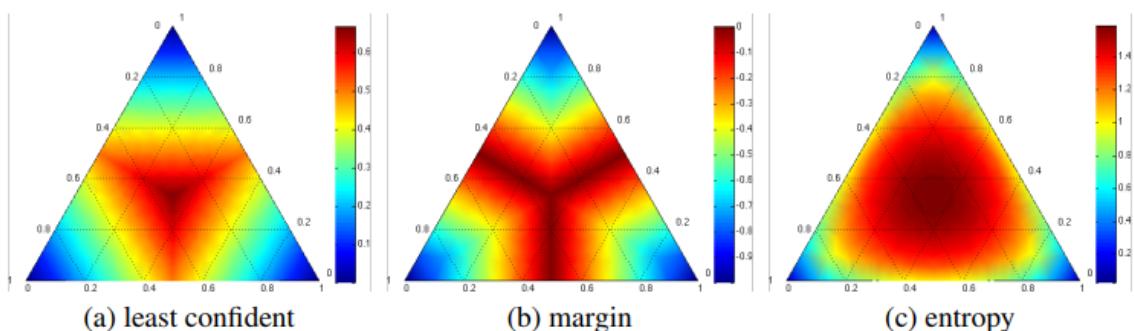
Một chiến lược lấy mẫu không chắc chắn tổng quát hơn (và có thể phổ biến nhất) là sử dụng **entropy** như một độ đo không chắc chắn:

$$x_H^* = \operatorname{argmax}_x \left(- \sum_i P_\theta(y_i|x) \log P_\theta(y_i|x) \right)$$

Với y_i là tất cả các lớp có thể có. *Entropy* là một độ đo lý thuyết thông tin đại diện cho lượng thông tin cần thiết để mã hoá một phân phối. *Entropy* thoả mãn các điều kiện sau:

- Entropy phải tỷ lệ thuận liên tục với các xác suất xuất hiện của các phần tử ngẫu nhiên trong tín hiệu. Thay đổi nhỏ trong xác suất phải dẫn đến thay đổi nhỏ trong entropy.
- Nếu các phần tử ngẫu nhiên đều có xác suất xuất hiện bằng nhau, việc tăng số lượng phần tử ngẫu nhiên phải làm tăng entropy.
- Có thể tạo các chuỗi tín hiệu theo nhiều bước, và entropy tổng cộng phải bằng tổng có trọng số của entropy của từng bước.

Do đó, nó thường được coi là độ đo không chắc chắn trong máy học. Đối với phân loại nhị phân, trên thực tế, cả 3 chiến lược trên đều tương đương với truy vấn trường hợp có xác suất hậu nghiệm gần với 0.5 nhất. Tuy nhiên, phương pháp dựa trên *entropy* dễ dàng tổng quát hoá thành bộ phân loại đa nhãn và các mô hình xác suất cho các cấu trúc phức tạp hơn như chuỗi (sequence) và cây (tree).



Hình 4.6: Heatmap minh họa hành vi truy vấn của các phương pháp lấy mẫu không chắc chắn phổ biến trong một bài toán phân loại ba nhãn.

Hình 4.6 mô tả mối quan hệ tiềm ẩn giữa các độ đo không chắc chắn này. Trong mọi trường hợp, mẫu thông tin nhất sẽ nằm ở tâm của tam giác, bởi vì điều này đại

diện cho phân bố xác suất các lớp gần như là đồng nhất (và do đó ít chắc chắn nhất trong mô hình). Tương tự, ít thông tin nhất ở 3 góc, khi đó độ không chắc chắn của mô hình là nhỏ.

Các chiến lược lấy mẫu không chắc chắn cũng có thể được sử dụng với bộ phân loại không xác suất. Một trong những công trình đầu tiên khám phá lấy mẫu không chắc chắn được sử dụng là bộ phân loại cây quyết định (decision tree) [14]. Cách tiếp cận tương tự được áp dụng cho học tích cực với bộ phân loại hàng xóm gần nhất (nearest-neighbor) [15], bằng cách cho phép mỗi hàng xóm vote nhãn lớp của x , với tỷ lệ phiếu đại diện cho xác xuất đầu ra. Tong và Koller (2000) đã thực nghiệm với lấy mẫu không chắc chắn cho support vector machines - SVM [16], liên quan đến việc truy vấn trường hợp gần nhất với ranh giới quyết định tuyến tính, điều này cũng tương tự với các tiếp cận hồi quy logistic hoặc Naive bayes.

Cho đến nay chúng ta mới chỉ thảo luận về các nhiệm vụ phân loại, nhưng lấy mẫu độ không chắc chắn cũng có thể áp dụng trong các bài toán hồi quy (tức là các nhiệm vụ học tập trong đó biến đầu ra là một giá trị liên tục chứ không phải là một tập hợp các nhãn lớp rời rạc). Trong cài đặt này, người học chỉ cần truy vấn các mẫu không được gắn nhãn mà mô hình có phương sai đầu ra cao nhất trong dự đoán của nó. Học tích cực cho các bài toán hồi quy đã có lịch sử lâu đời trong các tài liệu thống kê, thường được gọi là thiết kế thực nghiệm tối ưu [17]. Những cách tiếp cận như vậy tránh được việc lấy mẫu độ không chắc chắn thay cho các chiến lược phức tạp hơn.

4.3.2 Truy vấn dựa vào hội đồng

Một khung lựa chọn truy vấn khác là **truy vấn dựa vào hội đồng (query-by-committee) (QBC)** [18]. Phương pháp QBC bao gồm duy trì một nhóm các mô hình hội đồng $\mathcal{C} = \{\theta^{(1)}, \dots, \theta^{(2)}\}$ đã được huấn luyện trên tập nhãn hiện tại \mathcal{L} . Mỗi thành viên hội đồng sau đó được phép vote trên nhãn của các ứng cử viên truy vấn (vote cho các mẫu dữ liệu chưa có nhãn). Mẫu có thông tin truy vấn nhiều nhất được coi là trường hợp mà các mô hình bất đồng nhất.

QBC nhằm mục đích truy vấn trong các vùng gây tranh cãi của không gian mẫu đầu vào. Để mà triển khai thuật toán lựa chọn QBC, phải:

- i. Xây dựng một hội đồng các mô hình đại diện cho các vùng mẫu khác nhau của không gian các mẫu.
- ii. Có độ đo bất đồng giữa các thành viên trong hội đồng.

Để đo lường độ bất đồng, hai cách tiếp cận chính được đề xuất, đầu tiên là **vote entropy** [19]:

$$x_{VE}^* = \operatorname{argmax}_x \left(- \sum_i \frac{V(y_i)}{C} \log \frac{V(y_i)}{C} \right)$$

Với y_i là tất cả các lớp, và $V(y_i)$ là số vote nhận được từ thành viên hội đồng và C là kích cỡ của hội đồng. Đây có thể được coi là một QBC khái quát về lấy mẫu

không chắc chắn dựa trên entropy. Một biện pháp bất đồng khác đã được đề xuất là **Kullback-Leibler (KL) divergence** [20]:

$$x_{\text{KL}}^* = \underset{x}{\operatorname{argmax}} \frac{1}{C} \sum_{c=1}^C D(P_{\theta(c)} || P_c)$$

Với:

$$D(P_{\theta(c)} || P_c) = \sum_i P_{\theta(c)}(y_i|x) \log \frac{P_{\theta(c)}(y_i|x)}{P_c(y_i|x)}$$

Với $\theta^{(c)}$ đại diện cho một mô hình cụ thể trong hội đồng, và C đại diện cho toàn bộ hội đồng, do đó $P_c(y_i|x) = \frac{1}{C} \sum_{c=1}^C P_{\theta(c)}(y_i|x)$ là xác suất đồng ý y_i là nhãn đúng. *KL divergence* là một độ đo thông tin của sự khác nhau giữa hai phân phối xác suất. Vì vậy, độ đo bất đồng này coi là truy vấn nhiều thông tin nhất là mẫu có sự khác biệt trung bình lớn nhất giữa các phân phối nhãn của sự đồng thuận của hội đồng.

QBC cũng có thể được sử dụng trong bài toán hồi quy, tức là, bằng cách đo lường sự bất đồng dưới dạng phương sai giữa các dự đoán đầu ra của các thành viên hội đồng (Burbidge et al., 2007) [22].

Chương 5

Ứng dụng học tích cực cho bài toán phát hiện đối tượng

5.1 Học tích cực trong bài toán phát hiện đối tượng

Mô tả bài toán

Như đã đề cập ở chương 1, đồ án nghiên cứu phương pháp học tích cực cho bài toán phát hiện đối tượng. Đây là một trong những phương pháp học bán giám sát cho phát hiện đối tượng sử dụng học sâu. Giả sử dữ liệu ban đầu không có nhãn, khi đó mô hình khởi tạo sẽ là pretrained, tập dữ liệu có nhãn sẽ là rỗng. Tập dữ liệu không nhãn này sẽ được chia thành các phần ngẫu nhiên, nhằm đảm bảo phù hợp với tài nguyên máy tính để tính toán. Khi đó, ta sẽ tính điểm cho mỗi phần của tập dữ liệu không có nhãn này và lấy ra phần có điểm cao nhất yêu cầu người dùng gán nhãn. Khi đó mô hình sẽ huấn luyện các dữ liệu trong tập có nhãn và tập dữ liệu vừa được gán nhãn. Sau đó tập dữ liệu được gán nhãn trở thành dữ liệu có nhãn. Ta sẽ lặp lại cho đến khi tất cả các dữ liệu không có nhãn được gán nhãn hết.

Trong bài toán phát hiện đối tượng, gọi $v(x)$ là giá trị hoặc độ đo để đánh giá mẫu không có nhãn x . $v(x)$ cao có nghĩa là chúng ta nên chọn mẫu x cho việc gán nhãn. Mỗi ảnh có thể có nhiều đối tượng khác nhau, vì vậy chúng ta cần tổng hợp điểm trên các đối tượng trên ảnh để đánh giá điểm v cho ảnh đó. Có 3 phương pháp tổng hợp là: **sum, average, max**.

sum: Một phương pháp tổng hợp đơn giản nhất là lấy tổng (sum), theo đó, phương pháp này thích các hình ảnh mà có nhiều đối tượng không chắc chắn nhất. Khi sử dụng phương pháp này, các ảnh không có đối tượng sẽ có điểm là 0. Giá trị của một mẫu x có thể được tính từ các phát hiện D như sau:

$$v_{\text{sum}} = \sum_{i \in D} v(x_i)$$

trong đó, $v(x_i)$ là độ đo đánh giá đối tượng i trong mẫu ảnh đó.

average: Một phương pháp nữa là lấy trung bình của các điểm phát hiện. Nếu ảnh

không chứa đối tượng nào, nó sẽ được đánh giá điểm là 0.

$$v_{\text{average}} = \frac{1}{|D|} \sum_{i \in D} v(x_i)$$

max: Cuối cùng, chúng ta có thể đánh giá điểm v cho một ảnh bằng cách lấy max điểm các đối tượng trên ảnh. Phương pháp này có thể làm mất thông tin đáng kể, tuy nhiên, nó có lợi nếu ảnh có đối tượng gây nhiễu. Các ảnh không có đối tượng được coi như có điểm là 0. Công thức đánh giá ở phương pháp này như sau:

$$v_{\text{max}} = \max_{i \in D} v(x_i)$$

Thuật toán

Algorithm 1: Học tích cực cho phát hiện đối tượng

Require: Tập mẫu đã biết nhãn \mathcal{L} , tập mẫu chưa biết nhãn \mathcal{U} , mô hình khởi tạo f_0 , độ đo đánh giá học tích cực v ;

Require:

$f \leftarrow f_0$;

$\mathcal{U} = \mathcal{U}_1, \mathcal{U}_2, \dots \leftarrow$ Chia \mathcal{U} thành các batch ngẫu nhiên ;

while \mathcal{U} chưa rỗng **do**

Tính điểm cho tất cả các batch trong \mathcal{U} sử dụng f ;

$\mathcal{U}_{best} \leftarrow$ Batch có điểm cao nhất của \mathcal{U} theo v ;

$\mathcal{Y}_{best} \leftarrow$ Gán nhãn cho \mathcal{U}_{best} , thường là người gán nhãn ;

$f \leftarrow$ Huấn luyện f sử dụng \mathcal{L} và $(\mathcal{U}_{best}, \mathcal{Y}_{best})$;

$\mathcal{U} \leftarrow \mathcal{U} - \mathcal{U}_{best}$;

$\mathcal{L} \leftarrow \mathcal{L} \cup (\mathcal{U}_{best}, \mathcal{Y}_{best})$

end

5.2 Thu thập và biểu diễn dữ liệu

Dữ liệu được sử dụng trong báo cáo là bộ dữ liệu 3000 hình ảnh về súng ngắn Weapon (Nguồn: <https://github.com/ari-daszi/OD-WeaponDetection>). Đây là bộ dữ liệu mở, tập trung và hệ thống giám sát, an ninh. Một cách để ngăn chặn những tình huống xấu khi sử dụng súng là phát hiện ra chúng sớm trong những video giám sát.

Bộ dữ liệu gồm có 3000 ảnh về súng ngắn, mỗi một ảnh có một file *.xml tương ứng là nhãn của các đối tượng súng ngắn có trong ảnh. Để tương thích với hệ thống nhận dạng đối tượng YOLO, chúng ta cần chuyển đổi các thông số trong các file *.xml này về đúng định dạng như sau:

[ID class] [Hoành độ] [Tung độ] [Chiều dài] [Chiều rộng]

Nếu một ảnh có nhiều đối tượng thì mỗi hàng sẽ là một đối tượng, YOLO không tính lớp background giống như Faster-RCNN nên các ID class sẽ được đánh chỉ số từ 0. Toạ độ các hộp giới hạn phải được chuẩn hoá về đoạn [0, 1].

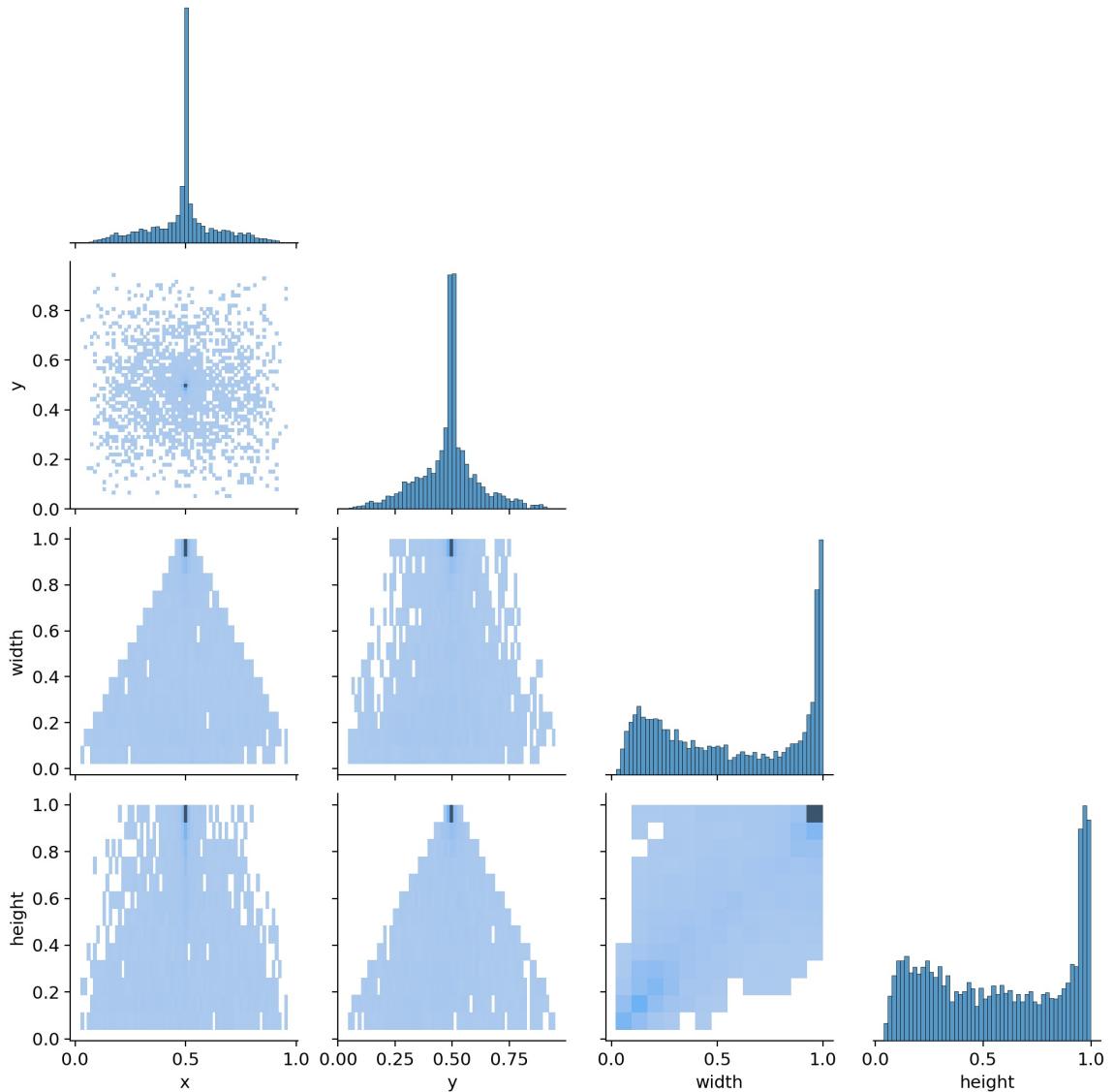
Dữ liệu được chia thành 2 phần, 80% được sử dụng cho huấn luyện và 20% được sử dụng cho kiểm thử.



Hình 5.1: Một mẫu ảnh trong bộ dữ liệu

```
| 0 0.385385 0.457275 0.263077 0.519630  
| 0 0.561538 0.239030 0.180000 0.468822
```

Hình 5.2: Nhãn tương ứng với mẫu ảnh hình 5.1



Hình 5.3: Tương quan của nhãn dữ liệu

5.3 Cấu hình YOLO

```
# parameters
nc: 1 # number of classes
depth_multiple: 0.33 # model depth multiple
width_multiple: 0.50 # layer channel multiple

# anchors
anchors:
- [10,13, 16,30, 33,23] # P3/8
- [30,61, 62,45, 59,119] # P4/16
- [116,90, 156,198, 373,326] # P5/32

# YOLO backbone
backbone:
# [from, number, module, args]
[[-1, 1, Focus, [64, 3]], # 0-P1/2
 [-1, 1, Conv, [128, 3, 2]], # 1-P2/4
 [-1, 3, BottleneckCSP, [128]],
 [-1, 1, Conv, [256, 3, 2]], # 3-P3/8
 [-1, 9, BottleneckCSP, [256]],
 [-1, 1, Conv, [512, 3, 2]], # 5-P4/16
 [-1, 9, BottleneckCSP, [512]],
 [-1, 1, Conv, [1024, 3, 2]], # 7-P5/32
 [-1, 1, SPP, [1024, [5, 9, 13]]],
 [-1, 3, BottleneckCSP, [1024, False]], # 9
]

# YOLO head
head:
[[-1, 1, Conv, [512, 1, 1]],
 [-1, 1, nn.Upsample, [None, 2, 'nearest']],
 [[-1, 6], 1, Concat, [1]], # cat backbone P4
 [-1, 3, BottleneckCSP, [512, False]], # 13

 [-1, 1, Conv, [256, 1, 1]],
 [-1, 1, nn.Upsample, [None, 2, 'nearest']],
 [[-1, 4], 1, Concat, [1]], # cat backbone P3
 [-1, 3, BottleneckCSP, [256, False]], # 17 (P3/8-small)

 [-1, 1, Conv, [256, 3, 2]],
 [[-1, 14], 1, Concat, [1]], # cat head P4
 [-1, 3, BottleneckCSP, [512, False]], # 20 (P4/16-medium)

 [-1, 1, Conv, [512, 3, 2]],
 [[-1, 10], 1, Concat, [1]], # cat head P5
 [-1, 3, BottleneckCSP, [1024, False]], # 23 (P5/32-large)

 [[17, 20, 23], 1, Detect, [nc, anchors]], # Detect(P3, P4, P5)
]
```

Hình 5.4: Cấu hình cho mô hình YOLO

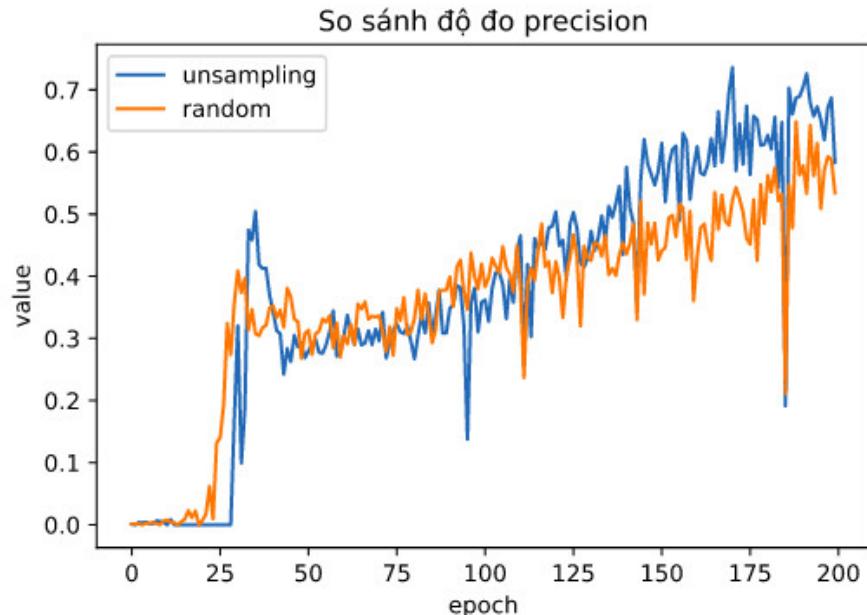
5.4 Kết quả thực nghiệm

Thông số cài đặt thực nghiệm

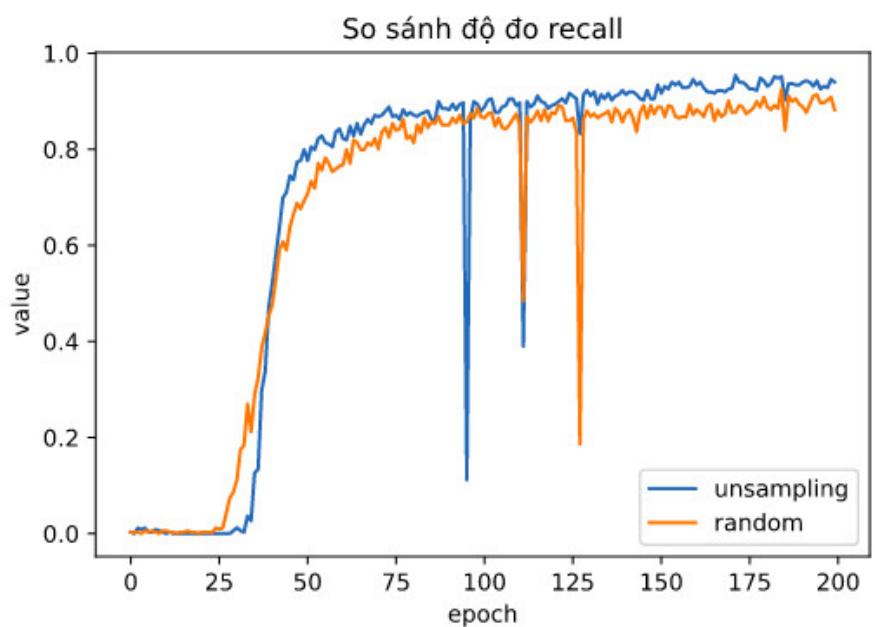
	Học tích cực	Phương pháp ngẫu nhiên
Kịch bản	Pool-based sampling	Pool-based sampling
Chiến lược truy vấn	Lấy mẫu không chắc chắn	Ngẫu nhiên
Phương pháp tổng hợp	Sum	Không sử dụng
Số ảnh 1 lần gán nhãn	8	8
Epoch huấn luyện mỗi truy vấn	2	2
Tổng số truy vấn	100	100

Bảng 5.1: Thông số cài đặt thực nghiệm

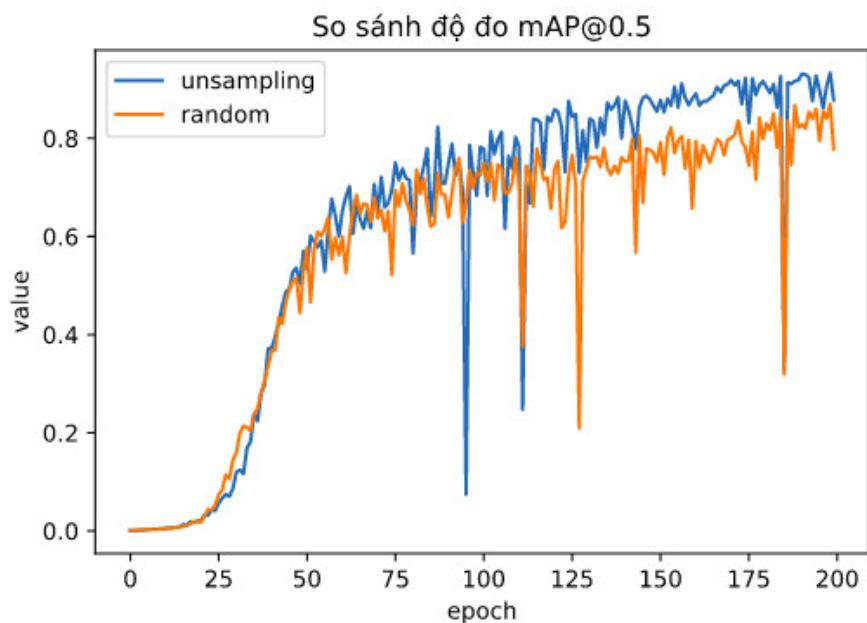
Một số đồ thị kết quả đánh giá trên các độ đo



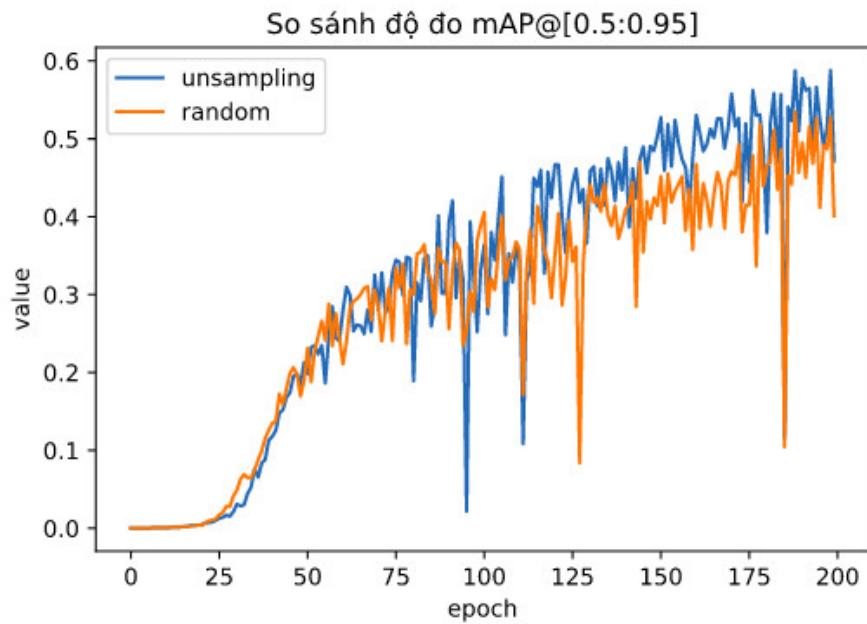
Hình 5.5: Đồ thị biểu diễn độ đo precision trên bộ dữ liệu kiểm thử



Hình 5.6: Đồ thị biểu diễn độ đo recall trên bộ dữ liệu kiểm thử

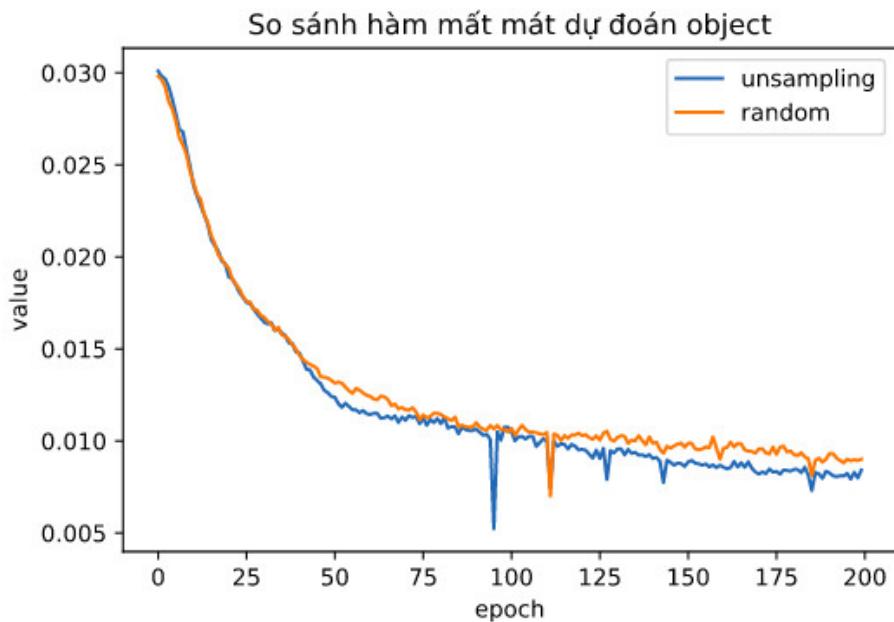


Hình 5.7: Đồ thị biểu diễn độ đo mAP@0.5 trên bộ dữ liệu kiểm thử

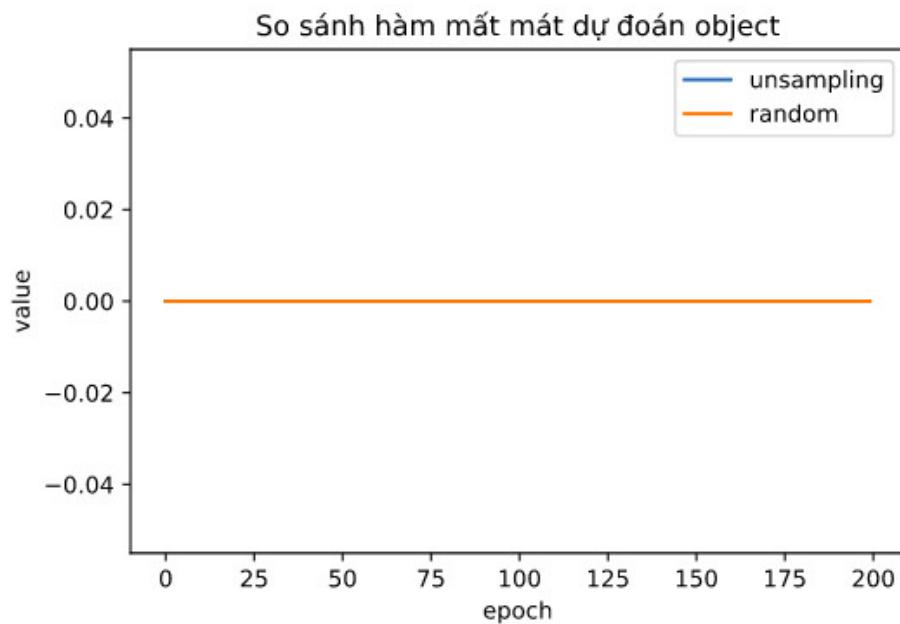


Hình 5.8: Đồ thị biểu diễn độ đo mAP@[0.5:0.05:0.95] trên bộ dữ liệu kiểm thử

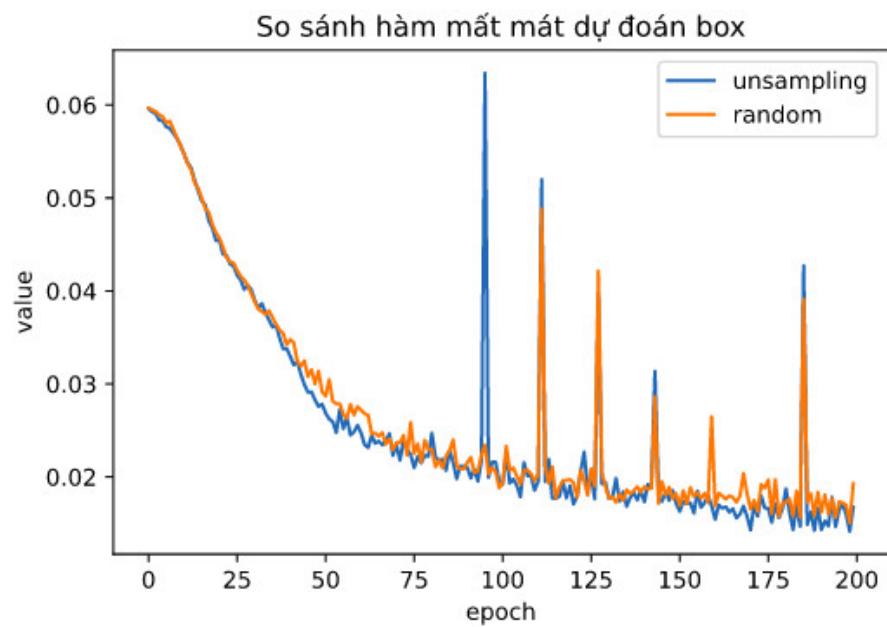
Các đồ thị biểu diễn sai số trên dữ liệu kiểm thử



Hình 5.9: Sai số dự đoán đối tượng



Hình 5.10: Sai số phân lớp. Do chỉ sử dụng một lớp dữ liệu nên sai số này luôn bằng 0.



Hình 5.11: Sai số định vị hộp giới hạn

Một số kết quả dự đoán



Hình 5.12: Một số kết quả dự đoán

KẾT LUẬN

Sau một thời gian thu thập tài liệu, khảo sát và phân tích nội dung, đồ án này là sự tổng hợp những nét chính trong học tích cực với bài toán phát hiện đối tượng. Sau đây là những điểm chính mà báo cáo đã giải quyết:

- Thu thập dữ liệu, tìm hiểu về các tài liệu liên quan, thử nghiệm một số phương pháp phát hiện đối tượng sử dụng học tích cực.
- Tìm hiểu phương pháp học tích cực, các trường hợp ứng dụng phù hợp, các thuật toán truy vấn trong học tích cực.
- Tìm hiểu phương pháp phát hiện đối tượng với mô hình học sâu YOLO và các mô hình cổ điển khác (Các mô hình khác không được lựa chọn vì tốc độ hoặc độ chính xác không đảm bảo cho vấn đề để gán nhãn và huấn luyện nhanh để tương tác với người dùng).
- Ứng học học tích cực trong bài toán phát hiện đối tượng trong ảnh. Với mô hình học thụ động, cần một lượng lớn dữ liệu, học tích cực giúp làm giảm lượng dữ liệu cần huấn luyện này.

Công việc nghiên cứu trong tương lai:

- Cải tiến thuật toán học tích cực sử dụng các truy vấn khác nhau.
- Xây dựng giao diện tương tác.

Tài liệu tham khảo

- [1] Redmon, J., Divvala, S., Girshick, R., Farhadi, "You only look once: Unified, real-time object detection ". In: CVPR. (2016).
- [2] Joseph Redmon and Ali Farhadi. "Yolo9000: better, faster, stronger". In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 7263–7271, 2017.
- [3] Joseph Redmon and Ali Farhadi. "Yolov3: An incremental improvement". CoRR, abs/1804.02767, 2018.
- [4] Bochkovskiy, A., Wang, C.Y., Liao, H.Y.M.: "Yolov4: Optimal speed and accuracy of object detection". arXiv preprint arXiv:2004.10934 (2020)
- [5] Settles, B. 2009. "Active learning literature survey". Computer Sciences Technical Report 1648, University of Wisconsin–Madison (2010).
- [6] Brust, C., Käding, C., Denzler, J.: "Active learning for deep object detection". CoRR abs/1809.09875 (2018).
- [7] Yao, A., Gall, J., Leistner, C., and Van Gool, L. 2012. "Interactive object detection". In Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on. IEEE, 3242–3249.
- [8] C. Lassner and R. Lienhart, “The fertilized forests Decision Forest Library,” in Proceedings of the 23rd Annual ACM Conference on Multimedia Conference (MM ’15), X. Zhou, A. F. Smeaton, Q. Tian, D. C. A. Bulterman, H. T. Shen, K. Mayer-Patel, and S. Yan, Eds. ACM, 2015, pp. 681–684.
- [9] Gall, V. Lempitsky, "Class-specific hough forests for object detection", in: CVPR, 2009
- [10] S. Ioffe and C. Szegedy. "Batch normalization: Accelerating deep network training by reducing internal covariate shift". arXiv preprint arXiv:1502.03167, 2015.
- [11] R. Girshick, J. Donahue, T. Darrell, and J. Malik. "Rich feature hierarchies for accurate object detection and semantic segmentation", In CVPR, 2014.

- [12] R. Girshick. "Fast R-CNN". arXiv:1504.08083, 2015.
- [13] S. Ren, K. He, R. Girshick, and J. Sun. "Faster R-CNN: Towards real-time object detection with region proposal networks", In NIPS, 2015.
- [14] D. Lewis and J. Catlett. "Heterogeneous uncertainty sampling for supervised learning". In Proceedings of the International Conference on Machine Learning (ICML), pages 148–156. Morgan Kaufmann, 1994.
- [15] M. Lindenbaum, S. Markovitch, and D. Rusakov. "Selective sampling for nearest neighbor classifiers". Machine Learning, 54(2):125–152, 2004.
- [16] S. Tong and D. Koller. "Support vector machine active learning with applications to text classification". In Proceedings of the International Conference on Machine Learning (ICML), pages 999–1006. Morgan Kaufmann, 2000.
- [17] V. Federov. *Theory of Optimal Experiments*. Academic Press, 1972.
- [18] H.S. Seung, M. Opper, and H. Sompolinsky. "Query by committee". In Proceedings of the ACM Workshop on Computational Learning Theory, pages 287–294, 1992.
- [19] Dagan and S. Engelson. "Committee-based sampling for training probabilistic classifiers". In Proceedings of the International Conference on Machine Learning (ICML), pages 150–157. Morgan Kaufmann, 1995.
- [20] A. McCallum and K. Nigam. "Employing EM in pool-based active learning for text classification". In Proceedings of the International Conference on Machine Learning (ICML), pages 359–367. Morgan Kaufmann, 1998.
- [21] D. Cohn, L. Atlas, and R. Ladner. "Improving generalization with active learning". Machine Learning, 15(2):201–221, 1994.
- [22] R. Burbidge, J.J. Rowland, and R.D. King. "Active learning for regression based on query by committee". In Proceedings of Intelligent Data Engineering and Automated Learning (IDEAL), pages 209–218. Springer, 2007.
- [23] K. Lang and E. Baum. "Query learning can work poorly when a human oracle is used". In Proceedings of the IEEE International Joint Conference on Neural Networks, pages 335–340. IEEE Press, 1992.
- [24] A. Fujii, T. Tokunaga, K. Inui, and H. Tanaka. "Selective sampling for examplebased word sense disambiguation". Computational Linguistics, 24(4):573–597, 1998.