

**BỘ GIÁO DỤC VÀ ĐÀO TẠO
ĐẠI HỌC CẦN THƠ
TRƯỜNG CÔNG NGHỆ THÔNG TIN & TRUYỀN THÔNG**

**NIÊN LUẬN CƠ SỞ
NGÀNH CÔNG NGHỆ THÔNG TIN**



Đề tài

**NHẬN DIỆN ĐỐI TƯỢNG TRONG THỜI GIAN
THỰC VỚI YOLO**

**Sinh viên: Nguyễn Lê Quốc Thịnh
Mã số: B2014615
Khóa: K46**

Cần Thơ, 12/2023

**BỘ GIÁO DỤC VÀ ĐÀO TẠO
ĐẠI HỌC CẦN THƠ
TRƯỜNG CÔNG NGHỆ THÔNG TIN & TRUYỀN THÔNG
KHOA CÔNG NGHỆ THÔNG TIN**

**NIÊN LUẬN CƠ SỞ
NGÀNH CÔNG NGHỆ THÔNG TIN**



Đề tài

NHẬN DIỆN ĐỐI TƯỢNG TRONG THỜI GIAN THỰC VỚI YOLO

**Người hướng dẫn
TS/Ths Phạm Thế Phi**

**Sinh viên thực hiện
Nguyễn Lê Quốc Thịnh
Mã số: B2014615
Khóa: K46**

Cần Thơ, 12/2023

LỜI CẢM ƠN

Lời đầu tiên, em xin gửi lời cảm ơn cảm ơn chân thành nhất đến GV. Phạm Thế Phi. Trong quá trình học tập và tìm hiểu và tìm hiểu môn Niên luận cơ sở ngành Công nghệ thông tin (CT271) , em đã nhận được sự quan tâm giúp đỡ, hướng dẫn rất tận tình tâm huyết của Thầy.

Do chưa có nhiều kinh nghiệm cũng như còn hạn chế về kiến thức, trong bài niên luận cơ sở chắc chắn sẽ không tránh khỏi những điều thiếu sót. Rất mong nhận được sự nhận xét, ý kiến đóng góp, phê bình từ Thầy để bài niên luận được hoàn thiện hơn.

Lời cuối cùng, em xin kính chúc Thầy nhiều sức khỏe, thành công và hạnh phúc trong cuộc sống cũng như công việc.

MỤC LỤC

PHẦN GIỚI THIỆU	8
1. BÀI TOÁN	8
2. MỤC TIÊU ĐỀ TÀI	8
3. ĐỐI TƯỢNG VÀ PHẠM VI NGHIÊN CỨU	8
3.1. Đối tượng nghiên cứu	8
3.2. Phạm vi nghiên cứu	8
4. PHƯƠNG PHÁP NGHIÊN CỨU	8
5. NỘI DUNG NGHIÊN CỨU	8
PHẦN NỘI DUNG	9
CHƯƠNG 1: ĐẶC TẢ VẤN ĐỀ	9
CHƯƠNG 2: CƠ SỞ LÝ THUYẾT	10
I. LÝ THUYẾT MẠNG TÍCH CHẬP (CNN)	10
1. Lý thuyết về các tầng	10
2. Các siêu tham số của bộ lọc	12
3. Điều chỉnh siêu tham số	14
4. Các hàm kích hoạt thường gặp	16
5. Phát hiện vật thể	17
6.YOLO	19
7. R-CNN	20
8. Xác nhận khuôn mặt và nhận diện khuôn mặt	20
9. Neural style transfer	22
10. Những kiến trúc sử dụng computational tricks	22
II. YOLO V4	24
1. Lý thuyết xây dựng mô hình	24
2. Kiến trúc tổng quan của mô hình	27
3. Kiến trúc của YOLO v4	28
CHƯƠNG 3: CÀI ĐẶT GIẢI PHÁP	31
1. Import các thư viện cần thiết	31

2. Tạo bản sao DarkNet cho YOLO V4	31
3. Cài đặt DarkNet cho YOLO v4 để có thể sử dụng	32
4. Tiến hành thử nghiệm YOLO v4 trên Webcam Video	33
CHƯƠNG 4: ĐÁNH GIÁ KIỂM THỬ	35
4.1.KIỂM THỬ	35
4.1.1. Mục tiêu kiểm thử	35
4.1.2. Kịch bản kiểm thử	35
4.1.3. Kết quả kiểm thử	36
4.2. ĐÁNH GIÁ	36
4.2.1. Quá trình kiểm thử	36
4.2.2. Kết quả kiểm thử	36
PHẦN KẾT LUẬN	38

DANH MỤC BẢNG

Bảng 1: Nội dung nghiên cứu	8
Bảng 2: Pooling	11
Bảng 3: Zero - padding	14
Bảng 4: Độ phức tạp của mô hình	16
Bảng 5: RLU	17
Bảng 6: Loại mô hình	18
Bảng 7: Mô hình xác nhận khuôn mặt và nhận diện khuôn mặt	21
Bảng 8: Kịch bản kiểm thử	36
Bảng 9: Kết quả kiểm thử	36

DANH MỤC HÌNH

Hình 1: Tổng quan kiến trúc	10
Hình 2: Lý thuyết tầng tích chập.....	10
Hình 3: Fully Conected	12
Hình 4: Các chiều của bộ lọc	12
Hình 5: Stride	13
Hình 6: Điều chỉnh tham số	15
Hình 7: Trường thụ cảm.....	16
Hình 8:NMS	19
Hình 9:YOLO.....	20
Hình 10:R-CNN	20
Hình 11:TRIPLET LOSS	22
Hình 12:Neural style transfer	22
Hình 13:GAN	23
Hình 14:Data Augmentation	24
Hình 15:IOU.....	26
Hình 16:Receptive Filed	27
Hình 17: Kiến trúc sinh thái.....	28
Hình 18: Cấu trúc tổng quan	30
Hình 19: So sánh độ chính xác.....	30
Hình 20: Cài đặt giải pháp 1	31
Hình 21: Cài đặt giải pháp 2	31
Hình 22: Cài đặt giải pháp 3	32
Hình 23: Cài đặt giải pháp 4	33
Hình 24: Luồng hoạt động	33
Hình 25: Màn hình sử dụng	34

TÓM LƯỢC

Trong thời đại công nghệ số hiện nay, việc ứng dụng trí tuệ nhân tạo và học sâu vào các lĩnh vực cuộc sống đã trở nên phổ biến hơn bao giờ hết. Trong dự án này, tôi áp dụng công nghệ nhận dạng đối tượng bằng YOLO để phát triển mô hình nhận dạng đối tượng trên thời gian thực. Mô hình này cho phép người dung có thể nhận dạng các loại đối tượng khác nhau trên thời gian thực tế.

Dự án sử dụng thư viện DarkNet làm nền tảng phát triển và trong quá trình phát triển tôi đã áp dụng thành công các công nghệ để hoàn thành mô hình nhận dạng. Tuy nhiên, do là dự án mới nên mô hình còn nhiều thiếu sót cần được cải tiến. Tôi rất mong nhận được những lời góp ý và phản hồi của người dung để ngày càng hoàn thiện dự án.

PHẦN GIỚI THIỆU

1. Bài toán:

Hiện nay xã hội công nghệ ngày càng phát triển và việc ứng dụng công nghệ thông tin vào đời sống là một nhu cầu vô cùng cấp bách. Trong đó nhận dạng đối tượng dựa vào trí tuệ nhân tạo vẫn còn là một thách thức. Việc nhận dạng đối tượng theo phương pháp trí tuệ nhân tạo có thể giúp tiết kiệm thời gian so với phương pháp thủ công truyền thống mất nhiều thời gian và luôn cần sự hỗ trợ của con người.

2. Mục tiêu đề tài:

- + Tìm hiểu kiến thức về Machine Learning
- + Tìm hiểu kiến thức về Object Detection
- + Tìm hiểu kiến thức về YOLO
- + Ứng dụng trong việc nhận dạng đối tượng trong thời gian thực

3. Đối tượng và phạm vi nghiên cứu:

3.1. Đối tượng nghiên cứu

Đối tượng nghiên cứu của đề tài là sử dụng Object Detection và YOLO trong việc nhận dạng đối tượng trong thời gian thực.

3.2. Phạm vi nghiên cứu

Phạm vi nghiên cứu chủ yếu của đề tài là sử dụng mô hình YOLO để nhận dạng đối tượng thông qua webcam trong thời gian thực.

4. Phương pháp nghiên cứu

Nội dung nghiên cứu chủ yếu từ những lý thuyết công nghệ có trên Paperswithcode và các tư liệu có sẵn trên Google. Từ đó vận dụng những kiến thức đã tiếp thu để thực hiện đề tài.

5. Nội dung nghiên cứu

Nội dung nghiên cứu được liệt kê ở đây:

STT	Đối tượng	Nội dung nghiên cứu
1	Mạng tích chập	Tìm hiểu lý thuyết về mạng tích chập
2	YOLO v4	Tìm hiểu lý thuyết về YOLO v4

Bảng 1: Nội dung nghiên cứu

PHẦN NỘI DUNG

CHƯƠNG 1: ĐẶC TẢ YÊU CẦU

Trong đời sống hiện nay, con người đã có những phát triển vượt bậc trong lĩnh vực khoa học công nghệ và đạt được nhiều thành tựu giúp cho đời sống con người ngày càng hiện đại, tiện dụng, dễ dàng. Lĩnh vực AI (trí tuệ nhân tạo) cũng là một trong những thành tựu đó, khi sự ra đời của AI đã giúp con người trong nhiều lĩnh vực như: Giao thông vận tải, y tế, giáo dục, dịch vụ,... . Thị giác máy tính (Computer Vision) là một trong những lĩnh vực của AI nhằm giúp máy tính có khả năng nhìn và hiểu được con người, bao gồm các phương pháp thu nhận, xử lý ảnh kỹ thuật số, phân tích và nhận dạng các hình ảnh. Thị giác máy tính được ứng dụng trong nhiều lĩnh vực như nhận diện con người, nhận diện sản phẩm, nhận diện biển số xe, nhận diện sản phẩm lỗi, nhận diện thương hiệu và tất cả các loại nhận diện vật thể. Nhận diện đối tượng cũng đã góp phần không nhỏ trong sự phát triển của nhân loại.

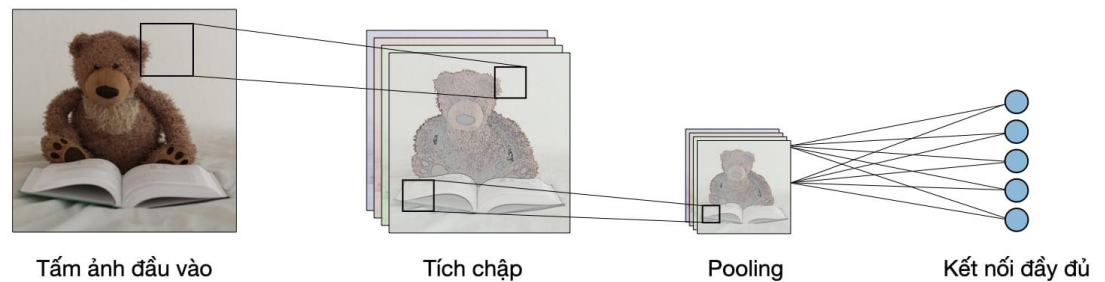
Nhận diện đối tượng (Object Detection) là một phần quan trọng trong lĩnh vực thị giác máy tính. Đây là quá trình sử dụng trí tuệ nhân tạo để xác định và phân loại các vật thể khác nhau trong hình ảnh hoặc video. Thay vì chỉ nhận ra một đối tượng, nhận diện đối tượng có khả năng xác định nhiều đối tượng khác nhau cùng một lúc, đồng thời gán nhãn và xác định vị trí của chúng.

Và với đề tài nghiên cứu “ Nhận diện đối tượng trong thời gian thực với YOLO “ có thể giúp chúng ta nhận diện vật thể trong thời gian thực tế giúp ích việc giám sát, bảo mật. Đồng thời còn giúp trong công tác điều tra phá án, khi giúp xác định được vật thể xuất hiện trong khung hình và còn rất nhiều tác dụng khác có thể ứng dụng trong đời sống.

CHƯƠNG 2: CƠ SỞ LÝ THUYẾT

I: Lý thuyết mạng tích chập (Convolutional Neural Network)

Tổng quan kiến trúc truyền thống được cấu thành bởi các tầng sau đây:

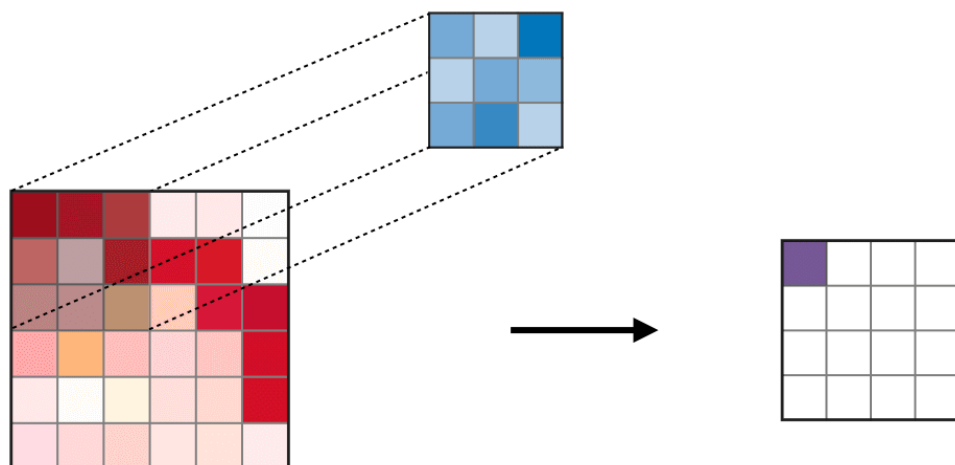


Hình 1: Tổng quan kiến trúc

1. Lý thuyết về các tầng:

1.1. Tầng tích chập (CONV)

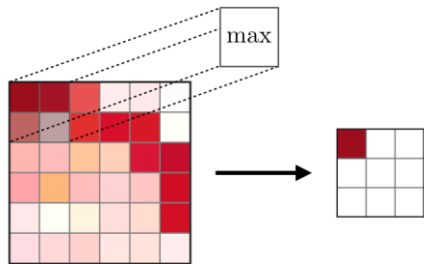
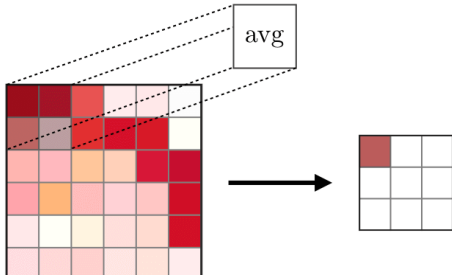
Tầng tích chập (CONV) sử dụng các bộ lọc để thực hiện phép tích chập khi đưa chúng đi qua đầu vào I theo các chiều của nó. Các siêu tham số của các bộ lọc này bao gồm kích thước bộ lọc F và độ trượt (stride) S . Kết quả đầu ra O được gọi là feature map hay activation map.



Hình 2: Lý thuyết tầng tích chập

1.2.Pooling (POOL)

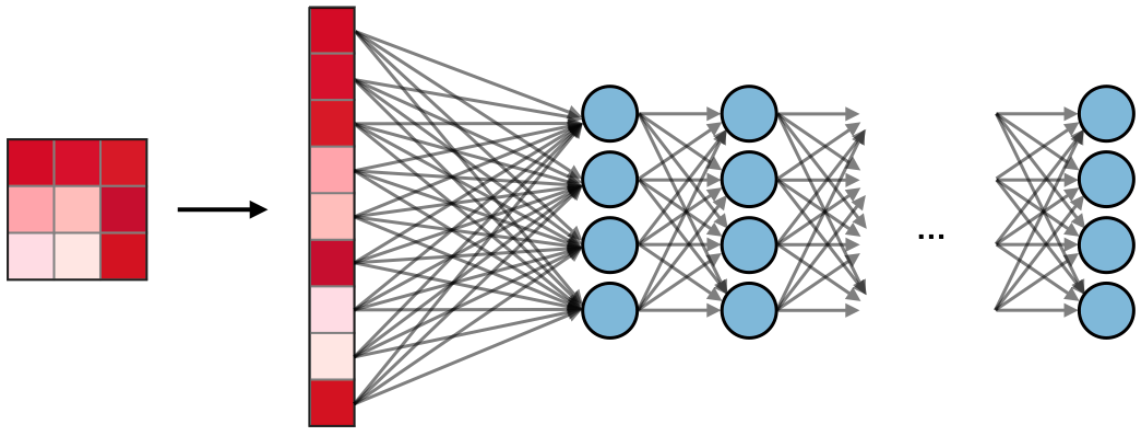
Tầng pooling (POOL) là một phép downsampling, thường được sử dụng sau tầng tích chập, giúp tăng tính bất biến không gian. Cụ thể, max pooling và average pooling là những dạng pooling đặc biệt, mà tương ứng là trong đó giá trị lớn nhất và giá trị trung bình được lấy ra.

Kiểu	Max pooling	Average pooling
Chức năng	Từng phép pooling chọn giá trị lớn nhất trong khu vực mà nó đang được áp dụng	Từng phép pooling tính trung bình các giá trị trong khu vực mà nó đang được áp dụng
Minh họa		
Nhận xét	<ul style="list-style-type: none"> -Bảo toàn các đặc trưng đã được phát hiện -Được sử dụng thường xuyên 	<ul style="list-style-type: none"> -Giảm kiến thức feature map -Được sử dụng trong mạng LeNet

Bảng 2: Pooling

1.3.Fully Connected (FC)

Tầng kết nối đầy đủ (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.



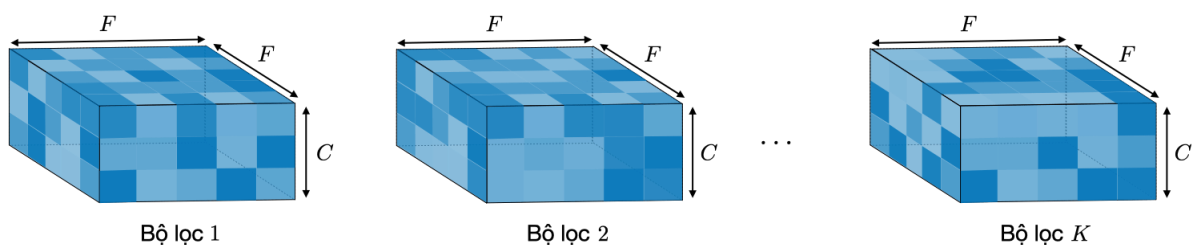
Hình 3: Fully Connected

2. Các siêu tham số của bộ lọc

Tầng tích chập chứa các bộ lọc mà rất quan trọng cho ta khi biết ý nghĩa đằng sau các siêu tham số của chúng.

2.1. Các chiều của một bộ lọc

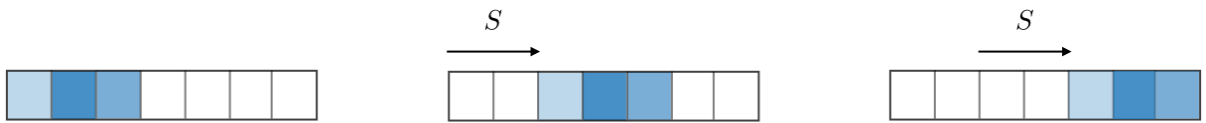
Một bộ lọc kích thước $F \times F$ áp dụng lên đầu vào chứa C kênh (channels) thì có kích thước tổng thể là $F \times F \times C$ thực hiện phép tích chập trên đầu vào kích thước $I \times I \times C$ và cho ra một feature map (hay còn gọi là activation map) có kích thước $O \times O \times 1$.



Hình 4: Các chiều của bộ lọc

2.2.Stride

Đối với phép tích chập hoặc phép pooling, độ trượt S ký hiệu số pixel mà cửa sổ sẽ di chuyển sau mỗi lần thực hiện phép tính.

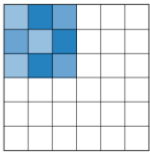
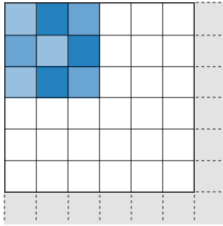
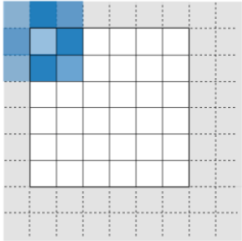


Hình 5: Stride

2.3.Zero-padding

Zero-padding là tên gọi của quá trình thêm P số không vào các biên của đầu vào. Giá trị này có thể được lựa chọn thủ công hoặc một cách tự động bằng một trong ba những phương pháp mô tả bên dưới:

Phương pháp	Valid	Same	Full
Giá trị	$P=0$	$P_{start} = \left\lceil \frac{S^l - I + F - S}{2} \right\rceil$ $P_{end} = \left\lceil \frac{S^l - I + F - S}{2} \right\rceil$	$P_{start} \in [0, F - 1]$ $P_{end} = F - 1$

Minh họa			
Mục đích	<ul style="list-style-type: none"> -Không sử dụng padding -Bỏ phép tích chập cuối nếu số chiều không khớp 	<ul style="list-style-type: none"> -Sử dụng padding để làm cho feature map có kích thước $\left\lceil \frac{I}{S} \right\rceil$ -Kích thước đầu ra phù hợp về mặt toán học -Còn được gọi là half padding 	<ul style="list-style-type: none"> -Padding tối đa sao cho phép tích chập có thể được sử dụng tại các rìa của đầu vào -Bộ lọc thấy được đầu vào từ đầu đến cuối

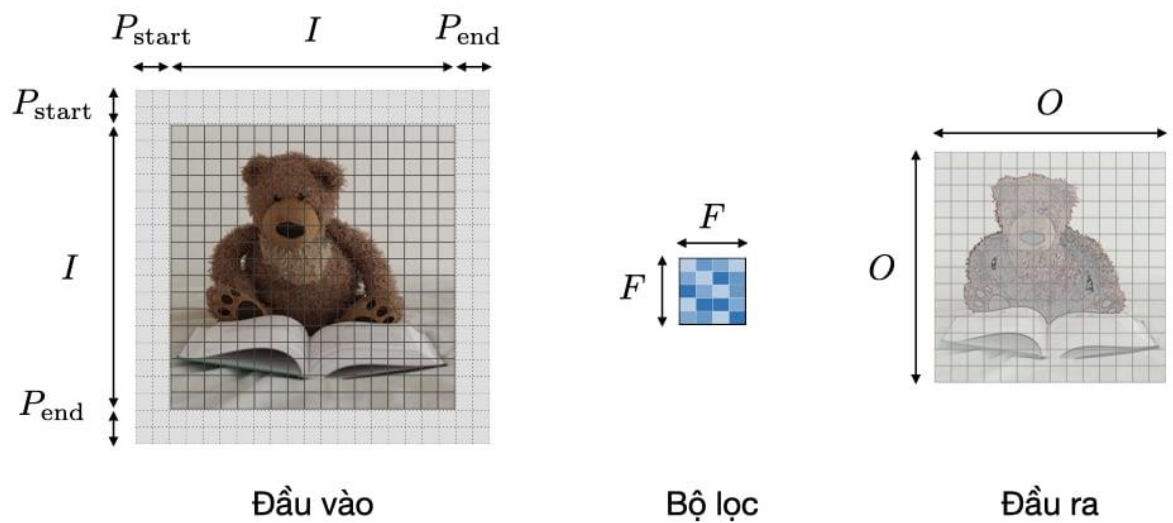
Bảng 3: Zero - padding

3. Điều chỉnh siêu tham số

3.1. Tính tương thích của tham số trong tầng tích chập

Bằng cách ký hiệu I là độ dài kích thước đầu vào, F là độ dài của bộ lọc, P là số lượng zero padding, S là độ trượt, ta có thể tính được độ dài O của feature map theo một chiều bằng công thức:

$$O = \frac{I - F + P_{start} + P_{end}}{S} + 1$$

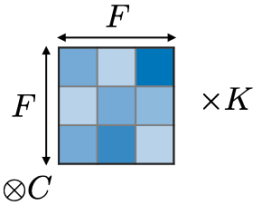
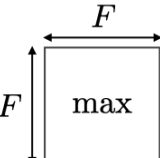
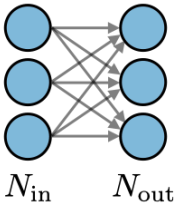


Hình 6: Điều chỉnh tham số

Lưu ý: Trong một số trường hợp, $P_{start} = P_{end} \triangleq P$, ta có thể thay thế $P_{start} + P_{end}$ bằng $2P$ trong công thức trên.

3.2. Hiểu về độ phức tạp của mô hình

Để đánh giá độ phức tạp của một mô hình, cách hữu hiệu là xác định số tham số mà mô hình đó sẽ có. Trong một tầng của mạng neural tích chập, nó sẽ được tính toán như sau:

	CONV	POOL	FC
Minh họa			
Kích thước đầu vào	$I \times I \times C$	$I \times I \times C$	N_{in}
Kích thước đầu ra	$O \times O \times K$	$O \times O \times C$	N_{out}

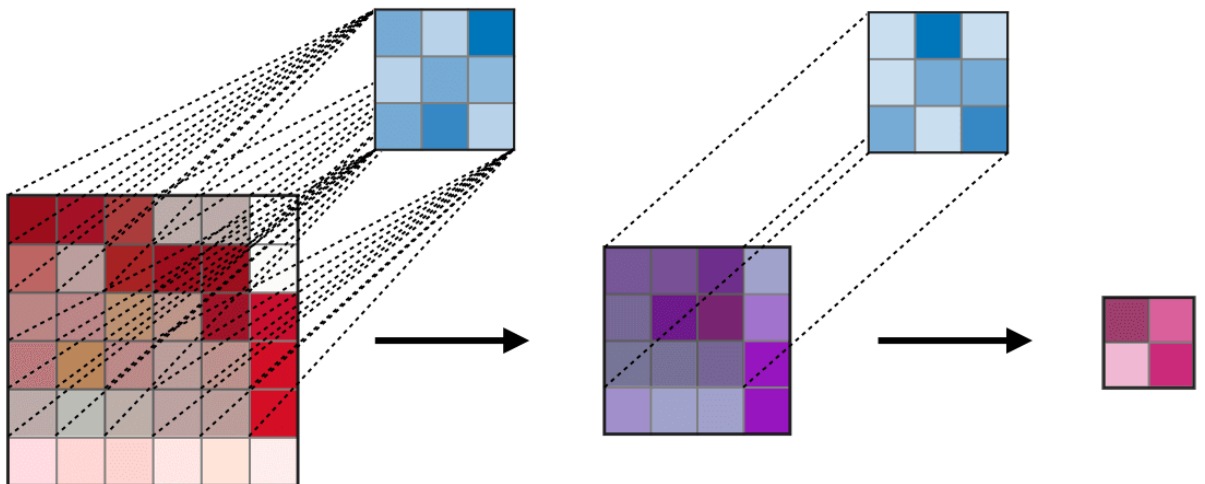
Số lượng tham số	$(F \times F \times C + 1) \cdot K$	0	$(N_{in} + 1) \times N_{out}$
Lưu ý	Mỗi tham số bias với mỗi bộ lọc Trong đa số trường hợp, $S < F$ Một lựa chọn phổ biến cho K là $2C$	Phép pooling được áp dụng lên từng kênh (channel-wise) Trong đa số trường hợp, $S = F$	Đầu vào được làm phẳng Mỗi neuron có một tham số bias Số neuron trong một tầng FC phụ thuộc vào ràng buộc kết cấu

Bảng 4: Độ phức tạp của mô hình

3.3. Trường thụ cảm

Trường thụ cảm (receptive field) tại tầng k là vùng được ký hiệu $R_k \times R_k$ của đầu vào mà những pixel của activation map thứ k có thể nhìn thấy. Bằng cách gọi F_j là kích thước bộ lọc của tầng J và S_i là giá trị độ trượt của tầng i và để thuận tiện., ta mặc định $S_0 = 1$, trường thụ cảm của tầng k được tính toán bằng công thức:

$$R_k = 1 + \sum_{j=1}^k F_j - 1 \prod_{i=0}^{j-1} S_i$$

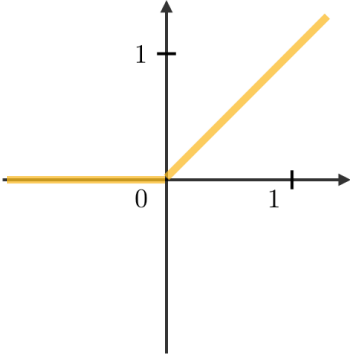
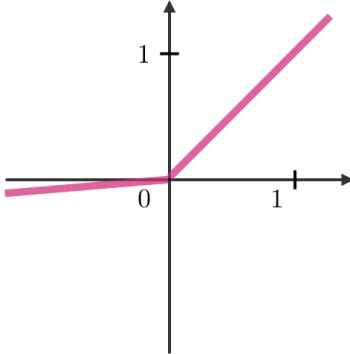
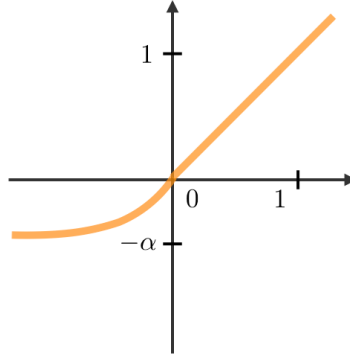


Hình 7: Trường thụ cảm

4. Các hàm kích hoạt thường gặp

4.1 Rectified Linear Unit

Tầng rectified linear unit (ReLU) là một hàm kích hoạt g được cho sử dụng trên tất cả các thành phần. Mục đích của nó là tăng tính phi tuyến tính cho mạng. Những biến thể khác của ReLU được tổng hợp ở bảng dưới:

ReLU	Leaky ReLU	ELU
$g(z) = \max(0, z)$	$g(z) = \max(\epsilon z, z)$ với $\epsilon \ll 1$	$g(z) = \max(\alpha(e^z - 1), z)$ với $\alpha \ll 1$
		
-Độ phức tạp phi tuyến tính có thể thông dịch được về mặt sinh học	-Gán vấn đề ReLU chết cho những giá trị âm	-Khả vi tại mọi nơi




Bảng 5: RLU

4.2. Softmax

Bước softmax có thể được coi là một hàm logistic tổng quát lấy đầu vào là một vector chứa các giá trị $x \in \mathbb{R}^n$ và cho ra là một vector gồm các xác suất $p \in \mathbb{R}^n$ thông qua một hàm softmax ở cuối kiến trúc.

5. Phát hiện vật thể (object detection)

5.1. Các kiểu mô hình

Phân loại hình ảnh	Phân loại cùng với khoanh vùng	Phát hiện
		
-Phân loại một tấm ảnh -Dự đoán xác suất của một vật thể	-Phát hiện một vật thể trong ảnh -Dự đoán xác suất của vật thể và định vị nó	-Phát hiện nhiều vật thể trong cùng một tấm ảnh -Dự đoán cùng xác suất của các vật thể và định vị chúng
CNN cổ điển	YOLO đơn giản hóa, R-CNN	YOLO, R-CNN

Bảng 6: Loại mô hình

5.2. Phát hiện

Trong bối cảnh phát hiện (detection) vật thể, những phương pháp khác nhau được áp dụng tùy thuộc vào liệu chúng ta chỉ muốn định vị vật thể hay phát hiện được những hình dạng phức tạp hơn trong tấm ảnh.

5.3. Intersection over Union

Tỉ lệ vùng giao trên vùng hợp, còn được biết đến là IoU, là một hàm định lượng vị trí Bp của hộp giới hạn dự đoán được vị trí đúng như thế nào so với hộp giới hạn thực tế Ba.

5.4.Anchor boxes

Hộp mờ neo 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 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 có khả năng là một hộp hình chữ nhật có hình dạng cho trước, trong khi dự đoán thứ hai sẽ là một hình hộp chữ nhật nữa với hình dạng hình học khác.

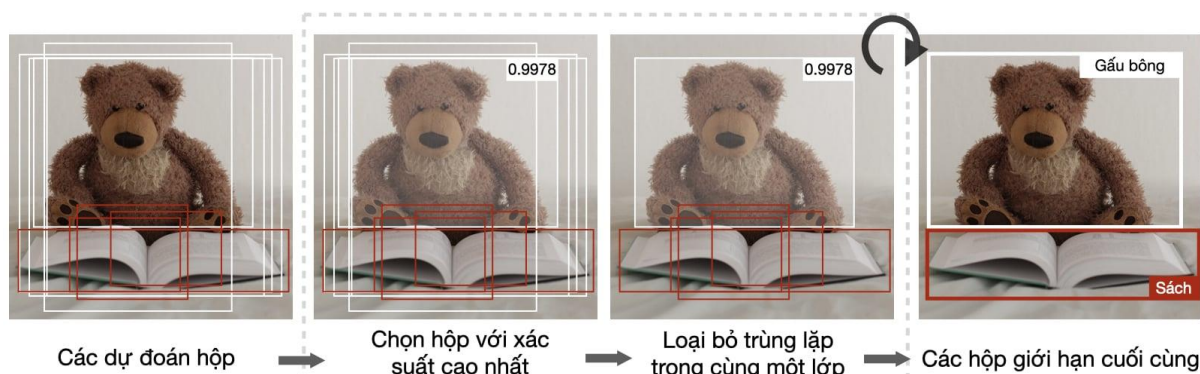
5.5.Non-max suppression

Kỹ thuật non-max suppression hướng tới việc loại bỏ những hộp giới hạn bị trùng chồng lên nhau của cùng một đối tượng bằng cách chọn chiếc hộp có tính đặc trưng nhất. Sau khi loại bỏ tất cả các hộp có xác suất dự đoán nhỏ hơn 0.6, những bước tiếp theo được lặp lại khi vẫn còn tồn tại những hộp khác.

Với một lớp cho trước

-Bước 1: Chọn chiếc hộp có xác suất dự đoán lớn nhất.

-Bước 2: Loại bỏ những hộp có $IoU \gg 0.5$ với hộp đã chọn.



Hình 8:NMS

6.YOLO

You Only Look Once (YOLO) là một thuật toán phát hiện vật thể thực hiện những bước sau:

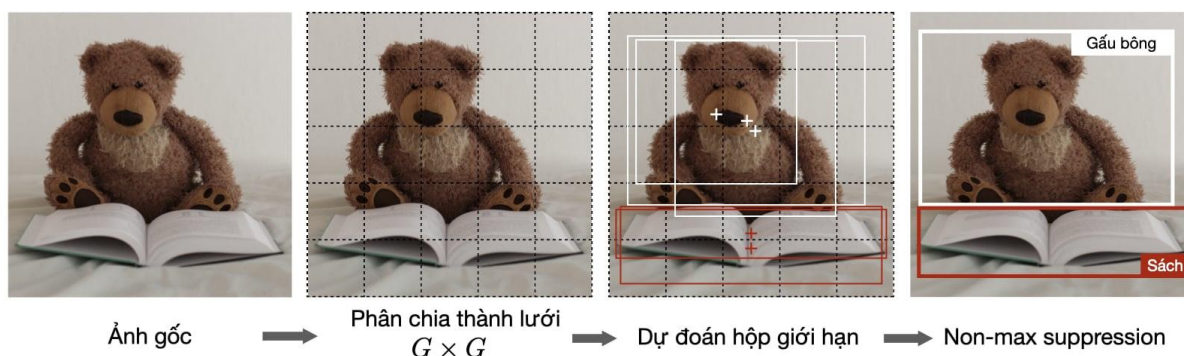
-Bước 1: Phân chia tấm ảnh đầu vào thành một lưới $G \times G$.

-Bước 2: Với mỗi lưới, chạy một mạng CNN dự đoán y có dạng sau:

$$Y = [p_c, b_x, b_y, b_h, b_w, c_1, c_2, \dots, c_p, \dots]^T \in R^{G \times G \times k \times (5+p)}$$

với p_c là xác suất dự đoán được một vật thể b_x, b_y, b_h, b_w là những thuộc tính của hộp giới hạn được dự đoán, c_1, \dots, c_p là biểu diễn one-hot của việc lớp nào trong p các lớp được dự đoán, và k là số lượng các hộp mờ neo.

-Bước 3: Chạy thuật toán non-max suppression để loại bỏ bất kỳ hộp giới hạn có khả năng bị trùng lặp.

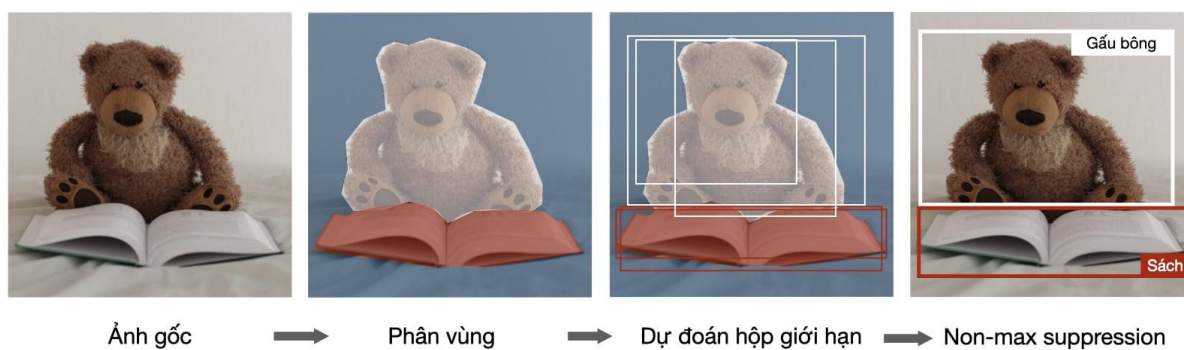


Hình 9:YOLO

Lưu ý: khi $p_c = 0$, thì mạng không phát hiện bất kỳ vật thể nào. Trong trường hợp, các dự đoán liên quan b_x, \dots, c_p sẽ bị bỏ đi.

7.R-CNN

Region with Convolutional Neural Networks (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 hộp giới hạn đó.



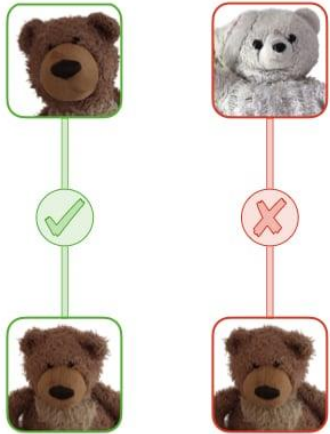

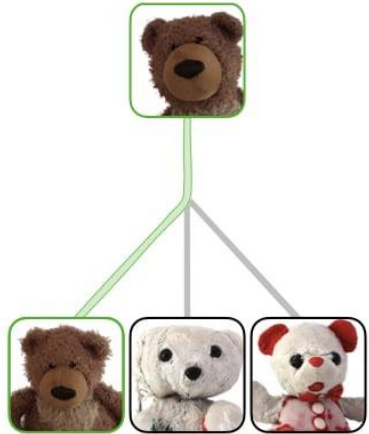

Hình 10:R-CNN

Lưu ý: mặc dù thuật toán gốc có chi phí tính toán cao và chậm, những kiến trúc mới đã có thể cho phép thuật toán này chạy nhanh hơn, như là Fast R-CNN và Faster R-CNN.

8.Xác nhận khuôn mặt và nhận diện khuôn mặt

8.1.Các kiểu mô hình :

Hai kiểu mô hình chính được tổng hợp trong bảng dưới:

Xác nhận khuôn mặt	Nhận diện khuôn mặt
-Có đúng người không ? -Tra cứu một-một	-Đây có phải là 1 trong K người trong cơ sở dữ liệu không ? -Tra cứu một với tất cả
<div> <div>Truy vấn</div>  </div> <div> <div>Tham vấn</div>  </div>	<div> <div>Truy vấn</div>  </div> <div> <div>Cơ sở dữ liệu</div>  </div>

Bảng 7: Mô hình xác nhận khuôn mặt và nhận diện khuôn mặt

8.2.One Shot Learning

One Shot Learning là một thuật toán xác minh khuôn mặt sử dụng một tập huấn luyện hạn chế để học một hàm similarity nhằm ước lượng sự khác nhau giữa hai tấm hình. Hàm này được áp dụng cho hai tấm ảnh thường được ký hiệu d.

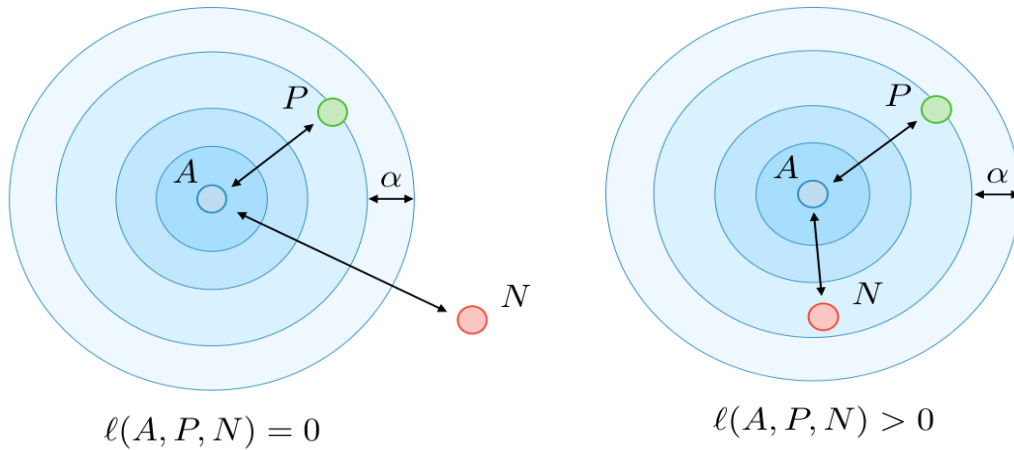
8.3.Siamese Network

Siamese Networks hướng tới việc học cách mã hóa tấm ảnh để rồi định lượng sự khác nhau giữa hai tấm ảnh. Với một tấm ảnh đầu vào $x^{(i)}$, đầu ra được mã hóa thường được ký hiệu là $f(x^{(i)})$.

8.4.Triplet loss

Triplet loss l là một hàm mất mát được tính toán dựa trên biểu diễn nhúng của bộ ba hình ảnh A (mỏ neo), P (dương tính) và N(âm tính). Ảnh mỏ neo và ảnh dương tính đều thuộc một lớp, trong khi đó ảnh âm tính thuộc về một lớp khác. Bằng cách gọi $\alpha \in R^+$ là tham số margin, hàm mất mát này được định nghĩa như sau:

$$l(A,P,N) = \max(d(A,P) - d(A,N) + \alpha, 0)$$



Hình 11:TRIPLET LOSS

9. Neural style transfer

Ý tưởng:

Mục tiêu của neural style transfer là tạo ra một ảnh G dựa trên một nội dung C và một phong cách S.



Hình 12:Neural style transfer

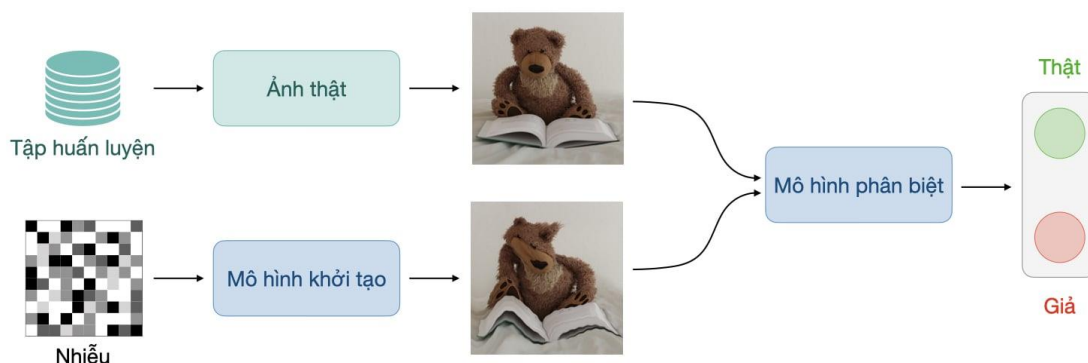
Tầng kích hoạt:

Trong một tầng l cho trước, tầng kích hoạt được kích hoạt $a^{[l]}$ và có các chiều là $n_H \times n_w \times n_c$.

10. Những kiến trúc sử dụng computational tricks

Generative Adversarial Network

Generative adversarial networks, hay còn được gọi là GAN, là sự kết hợp giữa mô hình khởi tạo và mô hình phân biệt, khi mà mô hình khởi tạo cố gắng tạo ra hình ảnh đầu ra chân thực nhất, sau đó được đưa vào mô hình phân biệt, mà mục tiêu của nó là phân biệt giữa ảnh được tạo và ảnh thật.



Hình 13:GAN

Lưu ý: có nhiều loại GAN khác nhau bao gồm từ văn bản thành ảnh, sinh nhạc và tổ hợp.

ResNet

Kiến trúc Residual Network (hay còn gọi là ResNet) sử dụng những khối residual blocks cùng với một lượng lớn các tầng để giảm lỗi huấn luyện. Những khối residual có những tính chất sau đây:

$$a^{[l+2]} = g(a^{[l]} + z^{[l+2]})$$

Inception Network

Kiến trúc này sử dụng những inception module và hướng tới việc thử các tầng tích chập khác nhau để tăng hiệu suất thông qua sự đa dạng của các feature. Cụ thể, kiến trúc này sử dụng thủ thuật tầng tích chập 1 X 1 để hạn chế gánh nặng tính toán.

II: Mô hình YOLO v4

Yolo v4 là mô hình được phát triển dựa trên ý tưởng của YOLO v3 nhưng được thêm những cải tiến mới như BoF, BoS để cải thiện độ chính xác và tốc độ của mô hình.

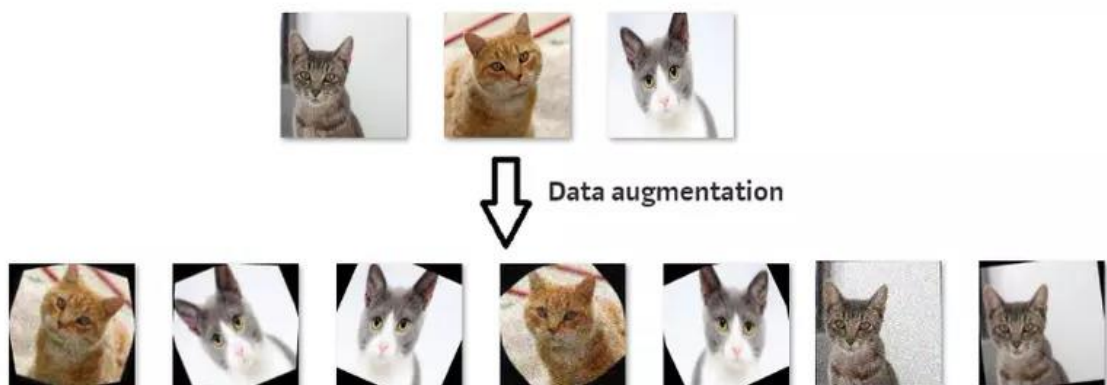
1.Lý thuyết xây dựng mô hình

1.1.Bag of Freebies (BoF)

Bag of Freebies là tập những kỹ thuật hoặc phương pháp mà thay đổi chiến thuật training hoặc chi phí training để có thể cải thiện độ chính xác của mô hình mà không làm tăng chi phí suy luận. Trong thực tế, phần lớn các mô hình CNN đều được huấn luyện offline nên điều này có thể được ứng dụng ở trong phần lớn các kiến trúc mô hình. Có rất nhiều phương pháp thay đổi chiến thuật training để cải thiện độ chính xác mô hình mà không làm tăng chi phí suy luận nhưng ở trong bài báo của YOLO v4, tác giả đưa ra 3 chiến thuật chính:

1.2.Data Augmentation

Data Augmentation là những phương pháp biến đổi ảnh đầu vào để làm gia tăng thêm độ phong phú của dữ liệu, từ đó model của chúng ta được huấn luyện trên những ảnh này có thể có khả năng chịu nhiễu tốt hơn. Có rất nhiều phương pháp Data Augmentation như thay đổi trục quang của ảnh: độ sáng, tương phản, hue, saturation,...; thay đổi hình học của ảnh: random scaling, cropping, flipping,...; hoặc có thể là vài phương pháp nâng cao như: Random Erase and CutOut, DropOut, Grid Mask,...; thậm chí có thể là kết hợp 2 ảnh bằng phương pháp: MixUp, CutMix, ...



Hình 14:Data Augmentation

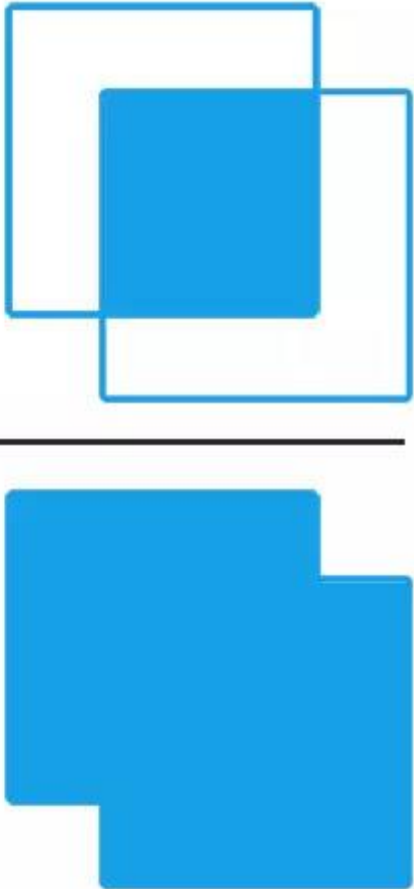
1.3.Semantic Distribution Bias in Datasets

Inherent bias là một mối quan ngại lớn vì nó đến từ sự phong phú trong tập dữ liệu dùng để train model. Nếu phân bố dữ liệu có vấn đề bias, nó có thể kéo quá trình training đến hội tụ đến cực tiểu địa phương và không thể khái quát hóa được. Nhìn chung vấn đề này từ 2 nguyên nhân chính từ tập dữ liệu:

- Data Imbalance giữa các classes
 - Trong mạng two-stage, vấn đề này có thể xử lý bằng cách tìm thêm những dữ liệu negative khó và dữ liệu online khó.
 - Trong mạng one-stage, người ta có thể dùng focal loss để xử lý vấn đề này.
- Không thể diễn giải mối quan hệ của mức độ liên kết giữa các lớp khác nhau với biểu diễn one-hot

1.4.Objective Function of BBox Regression

Trong bài toán Object Detection, hàm mục tiêu (hay còn gọi là hàm lỗi) phải cần được điều chỉnh để phù hợp với đầu ra của bài toán. Một phương pháp cổ điển là dùng hàm Mean Square Error (MSE) để tính toán giá trị sai khác giữa dự đoán của mô hình và ground truth của dữ liệu trên các giá trị là tọa độ điểm tâm (x_c , y_c) và chiều cao, chiều rộng của Bounding Box. Dĩ nhiên phương pháp này là không tốt, và người ta đưa ra 1 hướng tiếp cận khác là Anchor based approaches. Anchor based approaches là hướng tiếp cận sử dụng độ lệch tương ứng của những điểm này. Một phương pháp kinh điển nhất trong hướng tiếp cận này là IoU loss, tính toán độ trùng của box dự đoán và ground truth box.

$$\text{IoU} = \frac{\text{Area of Overlap}}{\text{Area of Union}}$$


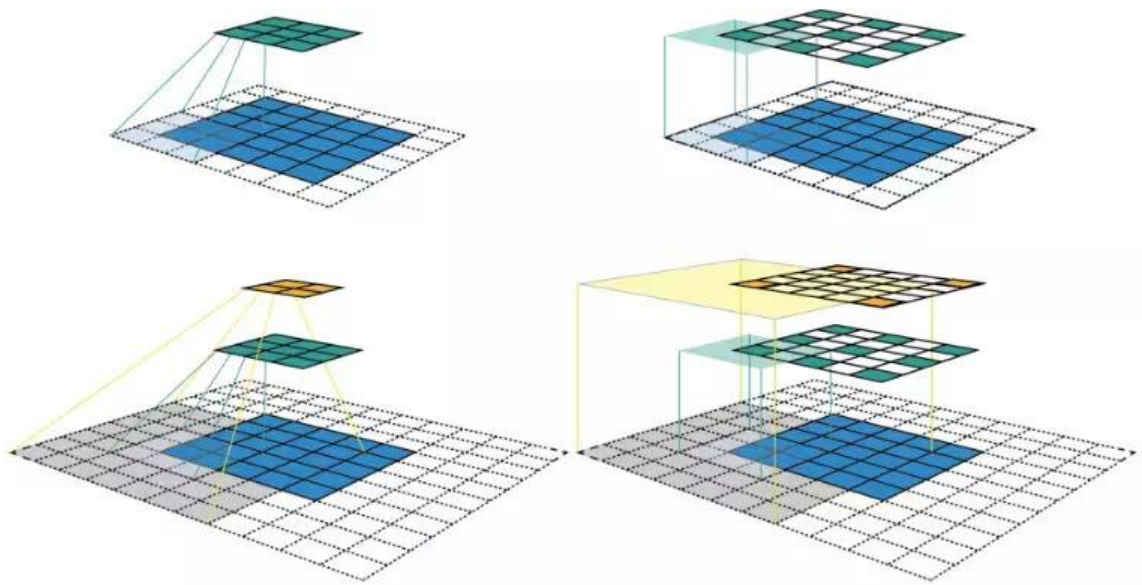
Hình 15:IOU

Bag of Speacials (BoS)

Khác với Bag of Freebies, Bag of Speacials là tập những phương pháp thêm một vài plugin modules và một vài post-processing methods để chỉ tăng thêm chi phí xử lý một chút nhưng cải thiện được khá độ chính xác của mô hình. Cụ thể hơn, những plugin modules này giúp tăng thêm một chút thuộc tính của model như mở rộng receptive field thêm cơ chế attention, tăng cường thêm khả năng tích hợp, ... và post-processing để hiển thị kết quả dự đoán của model (như hình vẽ ban đầu).

Receptive field

Receptive field là kích thước của một vùng trong không gian đầu vào được nhìn thấy bởi 1 pixel output. Không như mạng Fully Connected, một node có thể phụ thuộc vào toàn bộ input đầu vào của mạng thì trong CNN, 1 pixel output chỉ phụ thuộc vào 1 vùng của ảnh input, vùng này chính là receptive field.

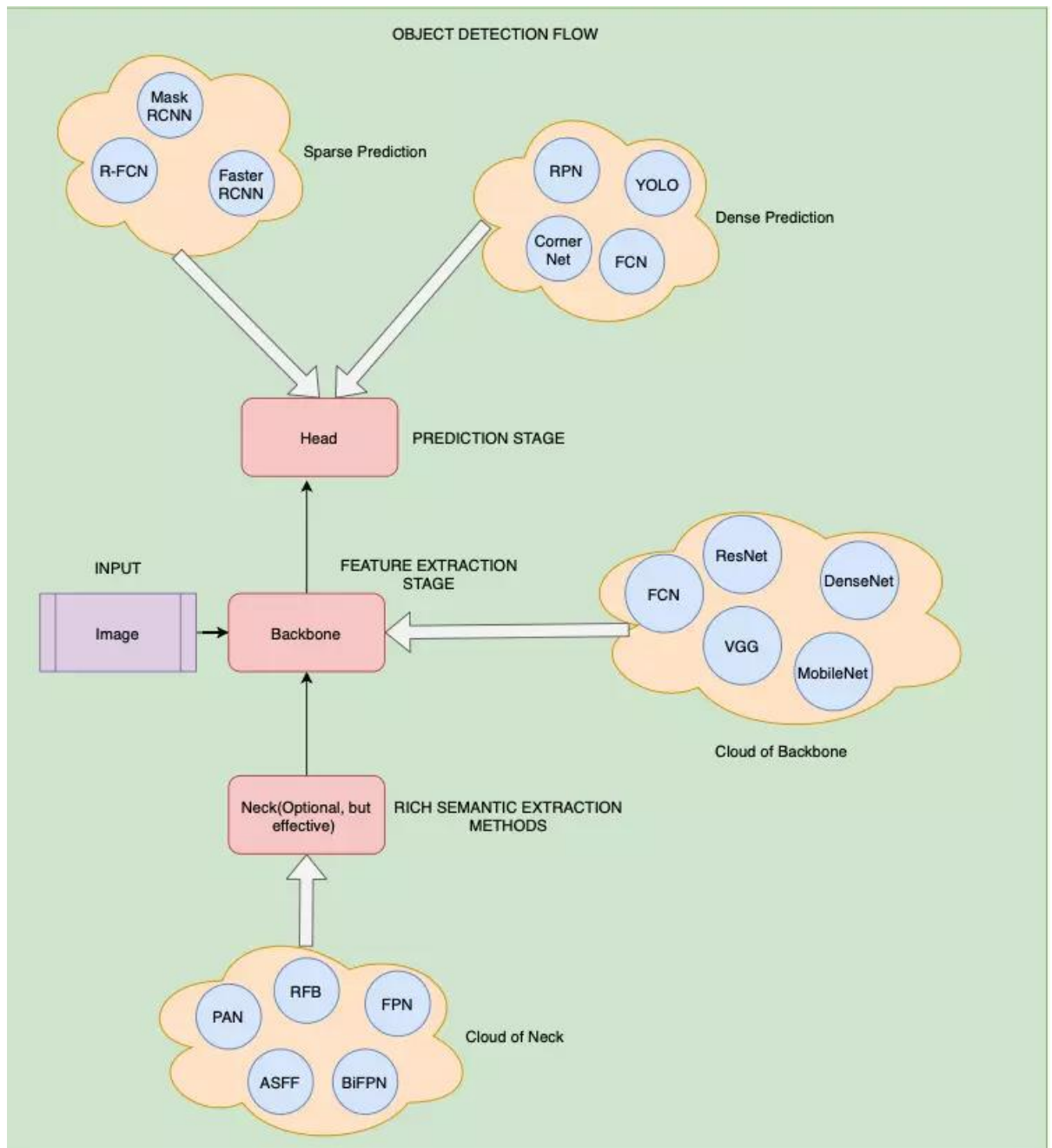


Hình 16:Receptive Filed

2.Kiến trúc tổng quan của một mô hình Object Detection

Nhìn chung một mô hình Object Detection được chia thành 2 thành phần chính là phần Backbone dùng cho feature extraction và một phần còn lại là phần Head dùng để dự đoán và tính toán giá trị lỗi mỗi lần dự đoán.

- Phần Backbone có thể sử dụng những mạng sẵn có được train trên tập ImageNet như VGG, ResNet, DenseNet, ... (nếu sử dụng trên những nền tảng có GPU) hoặc SqueezeNet, MobileNet, ShuffleNet (nếu sử dụng trên những nền tảng chỉ có CPU).
- Phần Head được chia làm 2 loại: one-stage và two-stage. Ngắn gọn thì two-stage là những kiểu mạng cổ điển mà quá trình ra output được chia làm 2 phần là localization và classification, trong khi đó one-stage thì gộp 2 quá trình này thành một. Vâng, và đúng là one-stage có khả năng dự đoán nhanh hơn rất nhiều so với two-stage object detection. Những mạng Two-stage phổ biến như : RCNN, FastRCNN, FasterRCNN,... còn những mạng one-stage phổ biến là: YOLO, SDD, RetinaNet,...
- Ngoài 2 phần trên ra thì trong những mạng hiện nay thì nhiều tác giả bổ sung thêm phần Neck. Phần này được thêm vào giữa Head và Backbone để tăng cường sự phong phú và khả năng biểu diễn ngữ nghĩa của các đối tượng được trích xuất cho các đối tượng có hình dạng và kích thước khác nhau. Hình ảnh dưới đây khái quát tổng quan về những thành phần trong mạng object detection.



Hình 17: Kiến trúc sinh thái

3. Kiến trúc của YOLO v4

Về kiến trúc thì YOLO v4 bao gồm 3 phần chính:

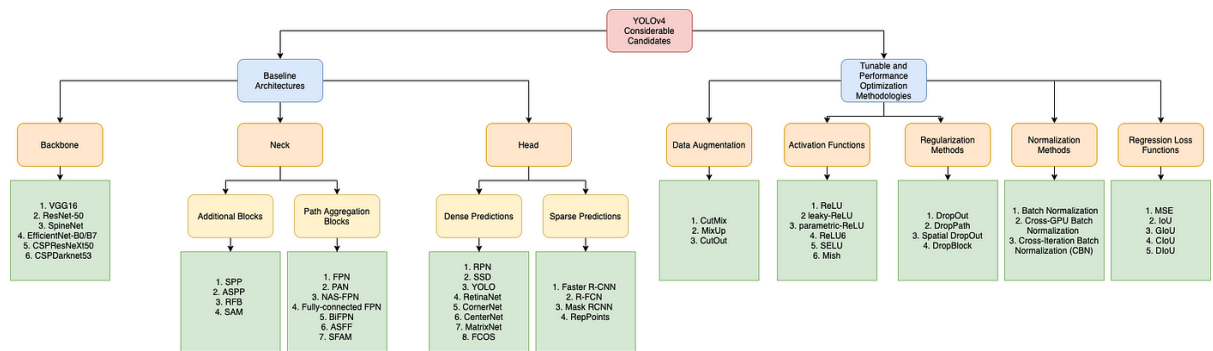
- Backbone : CSPDarknet53

- Neck: SPP (Spatial Pyramid Pooling), PAN (Path Aggregation Network)
- Head: YOLOv3

Ngoài ra, YOLO v4 còn sử dụng những BoF và BoS để tăng tốc quá trình training và cải thiện độ chính xác của mô hình như sau:

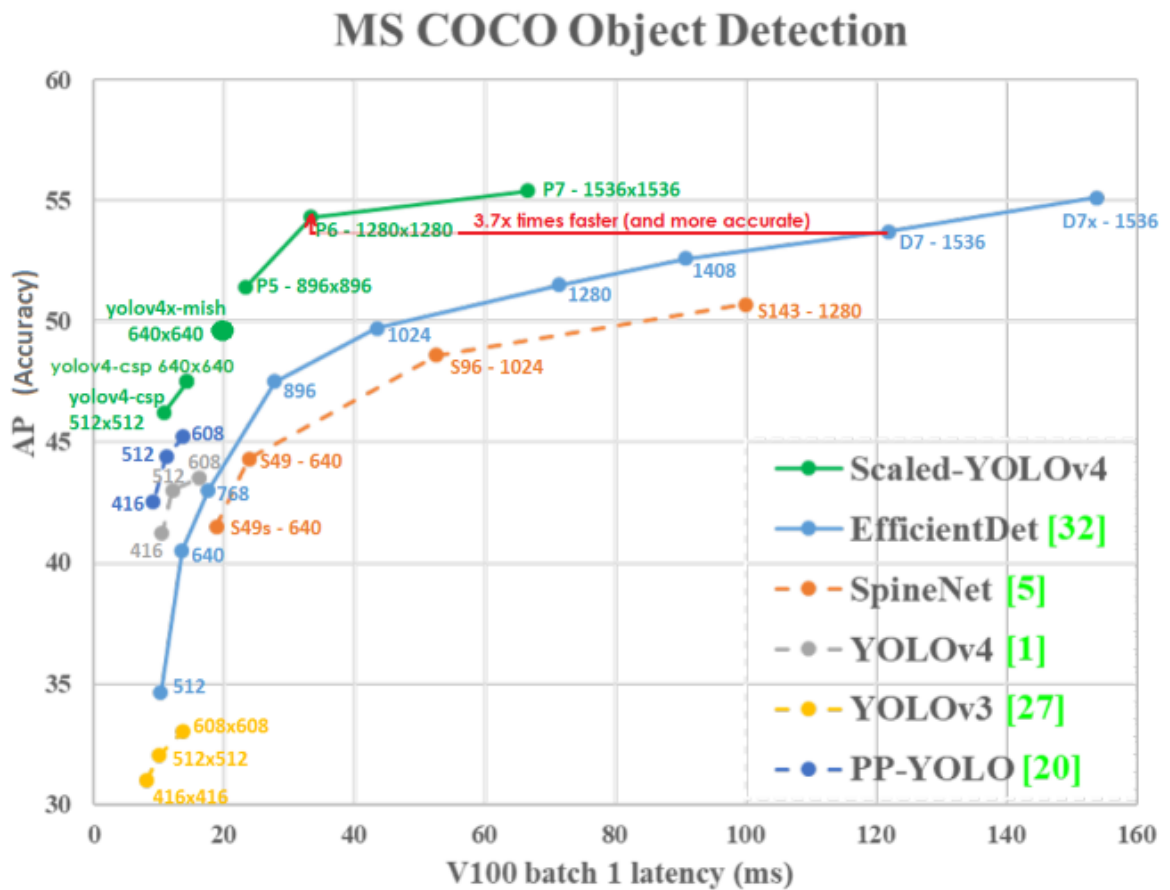
- **Bag of Freebies (BoF) sử dụng cho phần Backbone:**
 - CutMix và Mosaic data augmentation
 - DropBlock regularization
 - Class label smoothing
- **Bag of Specials (BoS) sử dụng cho phần Backbone:**
 - Mish activation
 - CSP block
 - Multi-input weighted residual connection
- **Bag of Freebies (BoF) sử dụng cho phần Detector:**
 - CIoU-loss
 - Cross mini Batch Normalization
 - DropBlock regularization
 - Mosaic data augmentation
 - Seft-Adversarial Training
 - Eliminate grid sensitivity
 - Multiple anchors
 - Cosine annealing scheduler
 - Optimal hyperparameters with Genetic Algorithm
- **Bag of Specials (BoS) sử dụng cho phần Detector:**
 - Mish activation
 - SPP-block
 - SAM-block
 - PAN
 - DIoU-NMS

Dưới đây là hình ảnh minh họa cấu trúc tổng quan của YOLO v4:



Hình 18: Cấu trúc tổng quan

Độ chính xác so với các loại model khác rất cao



Hình 19: So sánh độ chính xác

CHƯƠNG 3: CÀI ĐẶT GIẢI PHÁP

1: Import các thư viện cần thiết

```
[ ] #import dependencies
from IPython.display import display, Javascript, Image
from google.colab.output import eval_js
from google.colab.patches import cv2_imshow
from base64 import b64decode, b64encode
import cv2
import numpy as np
import PIL
import io
import html
import time
import matplotlib.pyplot as plt
%matplotlib inline
```

Hình 20: Cài đặt giải pháp 1

Các thư viện, gói xử lý cần có

-Display, Javascript, Image

-Eval_JS

-CV2_imshow

-CV2

-Numpy: Xử lý mảng đa chiều

-PIL

-IO

-HTML

-TIME

-Matplotlib.pyplot: Vẽ đồ thị

2: Cài đặt DarkNet cho YOLO v4

```
# clone darknet repo
!git clone https://github.com/AlexeyAB/darknet
```

```
# change makefile to have GPU, OPENCV and LIBSO enabled
%cd darknet
!sed -i 's/OPENCV=0/OPENCV=1/' Makefile
!sed -i 's/GPU=0/GPU=1/' Makefile
!sed -i 's/CUDNN=0/CUDNN=1/' Makefile
!sed -i 's/CUDNN_HALF=0/CUDNN_HALF=1/' Makefile
!sed -i 's/LIBSO=0/LIBSO=1/' Makefile
```

Hình 21: Cài đặt giải pháp 2


```
# make darknet (builds darknet so that you can then use the darknet.py file and have its dependencies)
!make
```

Lấy tệp trọng số yolov4 được chia tỷ lệ đã được đào tạo trước để phát hiện 80 lớp (đối tượng) từ ổ đĩa chung của Google Drive

```
# get the scaled yolov4 weights file that is pre-trained to detect 80 classes (objects) from shared google drive
!wget --load-cookies /tmp/cookies.txt "https://docs.google.com/uc?export=download&confirm=$(wget --quiet --save-cookies /tmp/cookies.txt --keep-session-cookies
```

3: Cài đặt DarkNet cho YOLO v4 để có thể sử dụng

```
[ ] # import darknet functions to perform object detections
from darknet import *
# load in our YOLOv4 architecture network
network, class_names, class_colors = load_network("cfg/yolov4-csp.cfg", "cfg/coco.data", "yolov4-csp.weights")
width = network_width(network)
height = network_height(network)

# darknet helper function to run detection on image
def darknet_helper(img, width, height):
    darknet_image = make_image(width, height, 3)
    img_rgb = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
    img_resized = cv2.resize(img_rgb, (width, height),
                              interpolation=cv2.INTER_LINEAR)

    # get image ratios to convert bounding boxes to proper size
    img_height, img_width, _ = img.shape
    width_ratio = img_width/width
    height_ratio = img_height/height

    # run model on darknet style image to get detections
    copy_image_from_bytes(darknet_image, img_resized.tobytes())
    detections = detect_image(network, class_names, darknet_image)
    free_image(darknet_image)
    return detections, width_ratio, height_ratio
```

Hình 22: Cài đặt giải pháp 3

Từ các hàm đã được định nghĩa trong file DarkNet.py, ta có thể sử dụng chúng trong YOLO v4 để tiến hành định nghĩa các khung của hình ảnh, tỷ lệ chiều ngang chiều dài sẽ xuất hiện trong khung hình

```
[ ] # function to convert the JavaScript object into an OpenCV image
def js_to_image(js_reply):
    """
    Params:
        js_reply: JavaScript object containing image from webcam
    Returns:
        img: OpenCV BGR image
    """
    # decode base64 image
    image_bytes = b64decode(js_reply.split(',')[1])
    # convert bytes to numpy array
    jpg_as_np = np.frombuffer(image_bytes, dtype=np.uint8)
    # decode numpy array into OpenCV BGR image
    img = cv2.imdecode(jpg_as_np, flags=1)

    return img

# function to convert OpenCV Rectangle bounding box image into base64 byte string to be overlayed on video stream
def bbox_to_bytes(bbox_array):
    """
    Params:
        bbox_array: Numpy array (pixels) containing rectangle to overlay on video stream.
    Returns:
        bytes: Base64 image byte string
    """
    # convert array into PIL image
    bbox_PIL = PIL.Image.fromarray(bbox_array, 'RGBA')
    iobuf = io.BytesIO()
    # format bbox into png for return
    bbox_PIL.save(iobuf, format='png')
```

```
def bbox_to_bytes(bbox_array):
    """
    Params:
        bbox_array: Numpy array (pixels) containing rectangle to overlay on video stream.
    Returns:
        bytes: Base64 image byte string
    """
    # convert array into PIL image
    bbox_PIL = PIL.Image.fromarray(bbox_array, 'RGBA')
    iobuf = io.BytesIO()
    # format bbox into png for return
    bbox_PIL.save(iobuf, format='png')
    # format return string
    bbox_bytes = 'data:image/png;base64,{}'.format(str(b64encode(iobuf.getvalue()), 'utf-8'))

    return bbox_bytes
```

Hình 23: Cài đặt giải pháp 4

Các hàm hỗ trợ xây dựng box nhận dạng vật thể

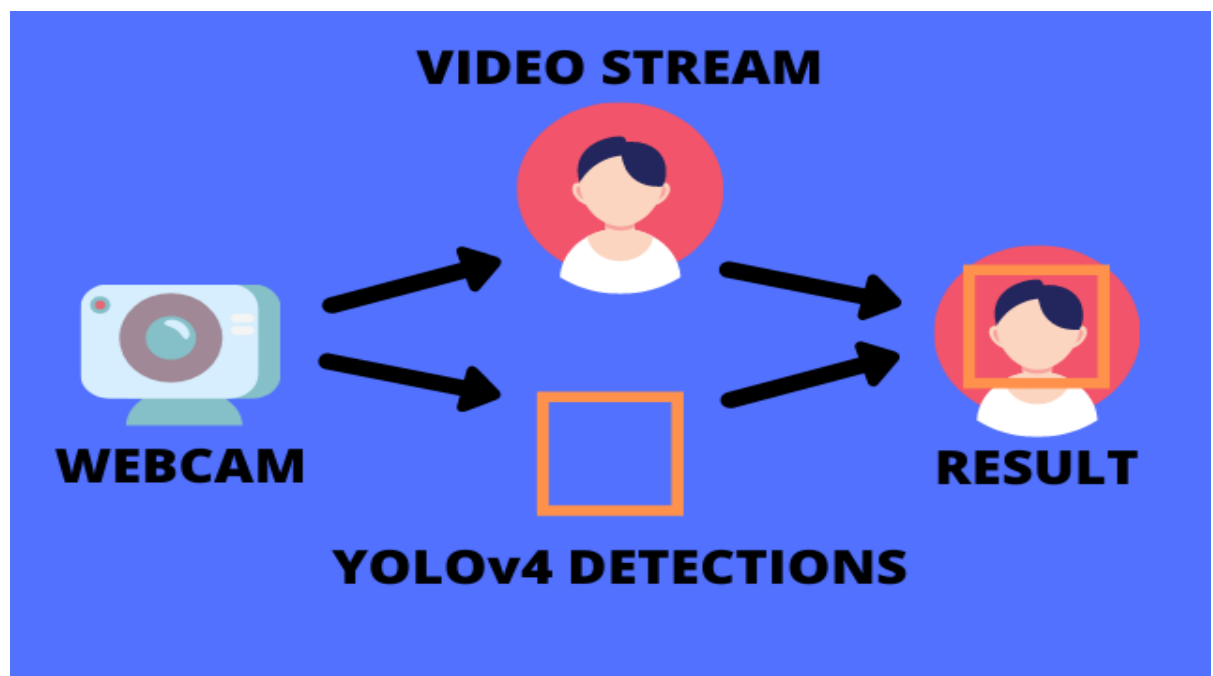
4: Tiến hành thử nghiệm YOLO v4 trên Webcam Video

Đầu vào: Webcam Video

Sau đó chạy mô hình YOLO v4 để phát hiện vật thể rồi tạo các hộp nhận dạng đối tượng

Đầu ra: hình ảnh hộp nhận dạng trên Webcam

Điểm đặc biệt của mô hình YOLO v4 là tốc độ vượt trội khiến cho mô hình có thể hoạt động ngay trên thời gian thực



Hình 24: Luồng hoạt động

Sử dụng JavaScript để tiến hành tạo luồng video trực tiếp bằng cách sử dụng Webcam



Hình 25: Màn hình sử dụng

CHƯƠNG 4: ĐÁNH GIÁ KIỂM THỬ

4.1 KIỂM THỬ

4.1.1. Mục tiêu kiểm thử

Kiểm thử nhằm đánh giá lại tính chính xác và tốc độ chạy mô hình trong thời gian thực và mức độ hoàn thành của mô hình, bên cạnh đó kiểm thử cũng nhằm mục đích phát hiện những lỗi còn sót lại, các lỗi phát sinh trong quá trình thử nghiệm để có biện pháp kịp thời góp phần đảm bảo khi đưa vào hoạt động sẽ diễn ra chính xác nhất, kiểm thử gồm các mục tiêu sau:

- Nhằm xác định các lỗi mắc phải
- Nhằm đảm bảo tính hoàn thiện của mô hình
- Trải nghiệm quá trình sử dụng

4.1.2. Kịch bản kiểm thử

Mã TestCase	Loại vật thể	Mô tả	Dữ liệu đầu vào	Kết quả mong đợi
TC01	Con người	Kiểm tra xem mô hình có nhận dạng được con người hay không ?	Hình ảnh con người xuất hiện trong khung hình Webcam	Hiển thị hộp nhận dạng với thông tin là con người
TC02	Điện thoại	Kiểm tra xem mô hình có nhận dạng được điện thoại hay không ?	Hình ảnh điện thoại xuất hiện trong khung hình Webcam	Hiển thị hộp nhận dạng với thông tin là điện thoại
TC03	Cái ly	Kiểm tra xem mô hình có nhận dạng được cái ly hay không?	Hình ảnh cái ly xuất hiện trong khung hình Webcam	Hiển thị hộp nhận dạng với thông tin là cái ly
TC04	Cây kéo	Kiểm tra xem mô hình có nhận dạng được cây kéo hay không?	Hình ảnh cây kéo xuất hiện trong khung hình Webcam	Hiển thị hộp nhận dạng với thông tin là cây kéo

TC05	Đồng hồ	Kiểm tra xem mô hình có nhận dạng được đồng hồ hay không?	Hình ảnh đồng hồ xuất hiện trong khung hình Webcam	Hiển thị hộp nhận dạng với thông tin là đồng hồ
------	---------	---	--	---

Bảng 8: Kịch bản kiểm thử

4.1.3. Kết quả kiểm thử

Mã TestCase	Kết quả mong đợi	Kết quả thực tế	Đánh giá
TC01	Hiển thị hộp nhận dạng với thông tin là con người	5 lần hiển thị thông tin là con người	5/5
TC02	Hiển thị hộp nhận dạng với thông tin là điện thoại	4 lần hiển thị thông tin là điện thoại 1 lần hiển thị là dây buộc	4/5
TC03	Hiển thị hộp nhận dạng với thông tin là cái ly	4 lần hiển thị thông tin là cái ly 1 lần hiển thị thông tin là ly rượu vang	4/5
TC04	Hiển thị hộp nhận dạng với thông tin là cái kéo	5 lần hiển thị thông tin là cái kéo	5/5
TC05	Hiển thị hộp nhận dạng với thông tin là đồng hồ	5 lần hiển thị thông tin là đồng hồ	5/5

Bảng 9: Kết quả kiểm thử

4.2.Đánh giá

4.2.1. Quá trình kiểm thử

Quá trình kiểm thử đã được thực hiện một cách nghiêm túc và đúng quy trình. Kế hoạch kiểm thử đã được tuân thủ đầy đủ và các trường hợp kiểm thử được thực hiện đầy đủ và chính xác. Kết quả kiểm thử được ghi lại chi tiết và minh bạch. Tổng thể quá trình kiểm thử đạt được các tiêu chuẩn cao và đóng góp tích cực cho dự án.

4.2.2.Kết quả kiểm thử

Trong quá trình kiểm thử, mô hình đã nhận dạng được các loại vật thể tuy nhiên vẫn còn các trường hợp nhận dạng sai. Tuy nhiên, đa phần các trường hợp đã được nhận dạng đúng và đảm bảo tính chính xác. Điều này cho thấy mô hình đã đáp ứng được tiêu chuẩn về độ chính xác và tin cậy.

PHẦN KẾT LUẬN

1.KẾT QUẢ ĐẠT ĐƯỢC:

Về lý thuyết và công nghệ:

- Mô hình đã ứng dụng thành công lý thuyết mạng tích chập vào việc xây dựng mô hình nhận dạng vật thể trong thời gian thực với độ chính xác cao.
- Hiểu rõ hơn về cách mô hình hoạt động
- Nâng cao khả năng đọc tài liệu giải quyết vấn đề
- Nâng cao kiến thức và kỹ năng

Về mô hình:

- Xây dựng thành công mô hình ứng dụng kỹ thuật cao

2.HẠN CHẾ:

Mô hình nhận dạng vật thể trong thời gian thực cũng có một số hạn chế như sau:

- Phạm vi nhận dạng còn bị bó hẹp
- Nhận diện sai một số vật thể
- Thời gian phát hiện chưa đủ nhanh

3.HƯỚNG PHÁT TRIỂN:

Thông qua những hạn chế, một số phương hướng phát triển đã được đề ra:

- Nâng cao phạm vi nhận dạng
- Cải tiến mô hình để đạt được thời gian phát hiện nhanh hơn

TÀI LIỆU THAM KHẢO

[1] Real Time Object Detection [Online]. Available:

<https://paperswithcode.com/task/real-time-object-detection>

[2] Mạng Neural tích chập cheatsheet [Online]. Available:

<https://stanford.edu/~shervine/1/vi/teaching/cs-230/cheatsheet-convolutional-neural-networks>

[3] YOLO v4 – kỹ nguyên cho những mô hình họ YOLO [Online]. Available:

<https://viblo.asia/p/yolov4-ky-nguyen-moi-cho-nhung-mo-hinh-ho-yolo-vlZL9N0dVQK>

PHỤ LỤC

Hướng dẫn sử dụng:

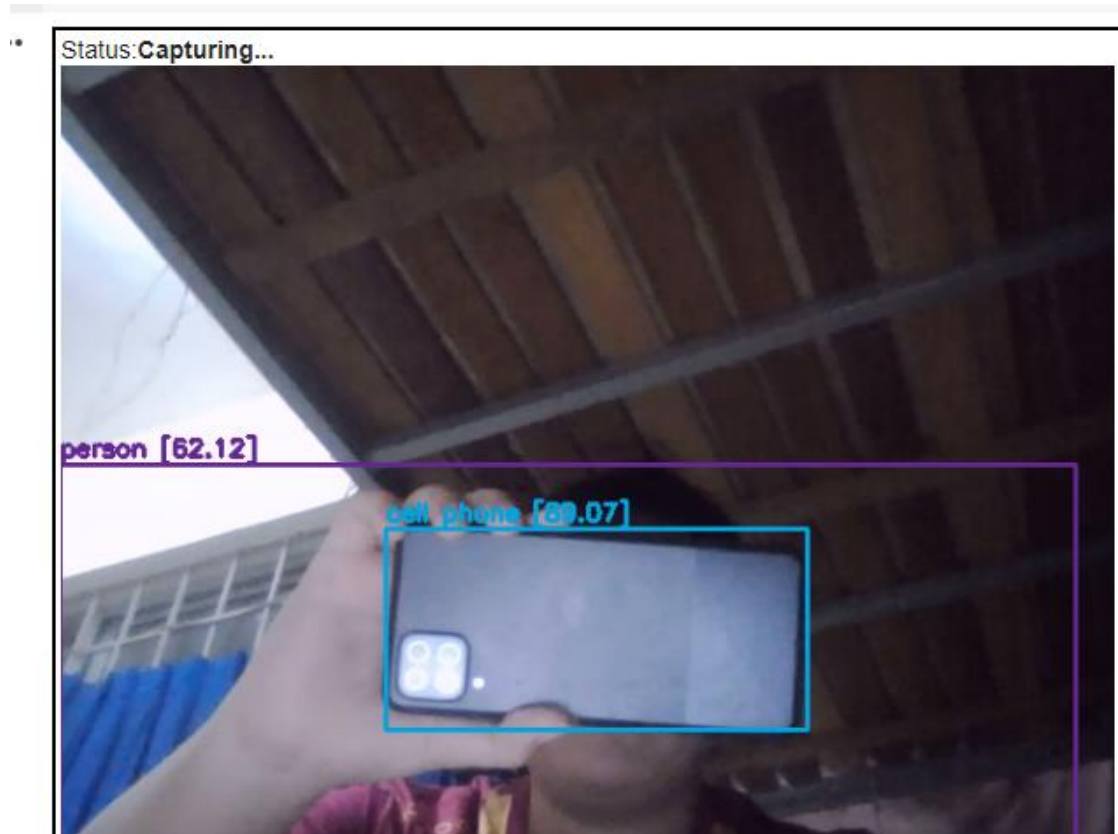
Người dùng bắt đầu bằng cách chạy tất cả các ô thực thi trong gg colab

Ví dụ:

```
✓ 0 # import dependencies
pầy from IPython.display import display, Javascript, Image
from google.colab.output import eval_js
from google.colab.patches import cv2_imshow
from base64 import b64decode, b64encode
import cv2
import numpy as np
import PIL
import io
import html
import time
import matplotlib.pyplot as plt
%matplotlib inline
```

Trạng thái ô đã được chạy xong

Sau đó có thể đưa các loại vật thể cần nhận dạng vào khung streaming



Khi đó mô hình có thể bắt đầu tiến hành nhận dạng

Khi người dùng muốn dừng quá trình nhận dạng thì click vào video để tiến hành ngừng streaming

