

DẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH  
TRƯỜNG ĐẠI HỌC BÁCH KHOA  
KHOA KHOA HỌC & KỸ THUẬT MÁY TÍNH



BÁO CÁO CUỐI KỲ  
ĐỒ ÁN TỔNG HỢP HƯỚNG TRÍ TUỆ NHÂN TẠO

*SỬ DỤNG DEEP LEARNING*

*NHẬN DIỆN PHƯƠNG TIỆN GIAO THÔNG*

Giảng viên hướng dẫn: Võ Thanh Hùng  
Sinh viên thực hiện: Nguyễn Lê Khanh - 2013444

Tp. Hồ Chí Minh, Tháng 12/2023



## MỤC LỤC

<b>1 GIỚI THIỆU ĐỀ TÀI</b>	<b>3</b>
1.1 Vấn đề bài toán đang giải quyết . . . . .	3
1.2 Ý tưởng giải quyết bài toán . . . . .	4
<b>2 GIẢI THUẬT YOLO V3</b>	<b>5</b>
2.1 Sơ lược về nhận diện vật thể - Object Detection . . . . .	5
2.2 Tổng quan về Convolutional Neural Network(CNN) . . . . .	6
2.2.1 Giới thiệu về CNN . . . . .	6
2.2.2 CNN hoạt động ra sao? . . . . .	6
2.3 YOLO V3 là gì? . . . . .	8
2.4 Cách hoạt động của YOLO V3 . . . . .	9
2.4.1 Input . . . . .	9
2.4.2 Kiến trúc mạng CNN của YOLO V3 . . . . .	9
2.4.3 Output . . . . .	13
2.4.4 Loss function . . . . .	14
2.4.5 Luồng xử lí của YOLO V3 . . . . .	15
2.5 Điểm mới của YOLO V3 so với các phiên bản trước đó . . . . .	17
<b>3 HIỆN THỰC MÔ HÌNH</b>	<b>19</b>
3.1 Dữ liệu huấn luyện mô hình . . . . .	19
3.2 Một số thông số của Model Summary . . . . .	19
3.2.1 Precision (P) . . . . .	19
3.2.2 Recall (R) . . . . .	19
3.2.3 Precision Recall Curve . . . . .	19
3.2.4 Average Precision (AP) . . . . .	20
3.2.5 mAP . . . . .	20
3.3 Tổng hợp những lần training . . . . .	20
3.3.1 Lần 1 . . . . .	20
3.3.2 Lần 2 . . . . .	21
3.3.3 Lần 3 . . . . .	22
<b>4 NHẬN XÉT MÔ HÌNH VÀ HƯỚNG PHÁT TRIỂN</b>	<b>24</b>
4.1 Những ưu và nhược điểm trong mô hình . . . . .	24
4.1.1 Ưu điểm . . . . .	24
4.1.2 Nhược điểm . . . . .	24
4.2 Hướng phát triển trong tương lai . . . . .	24



## DANH SÁCH HÌNH VẼ

<b>1.1</b>	Hình ảnh tình trạng kẹt xe diễn ra ở TP.Hồ Chí Minh . . . . .	3
<b>2.1</b>	Giai đoạn 1 . . . . .	5
<b>2.2</b>	Giai đoạn 2 . . . . .	6
<b>2.3</b>	One stage detectors . . . . .	6
<b>2.4</b>	Mô hình kiến trúc CNN điển hình. Nguồn ảnh: <a href="#">Wikipedia - Convolutional neural network</a>	7
<b>2.5</b>	Max pooling với bộ lọc 2x2 và bước nhảy 2. Nguồn ảnh: <a href="#">Wikipedia - Convolutional neural network</a> . . . . .	7
<b>2.6</b>	Average pooling với bộ lọc 2x2 và bước nhảy 2. Nguồn ảnh: <a href="#">Average Pooling</a> . . . . .	8
<b>2.7</b>	Fully-connection layer. Nguồn ảnh: <a href="#">Mạng neural tích chập cheatsheet</a> . . . . .	8
<b>2.8</b>	Mô hình thực hiện YOLO V3 . . . . .	9
<b>2.9</b>	DarkNet - 53. Nguồn ảnh: <a href="#">Tomato Diseases and Pests Detection Based on Improved Yolo V3 Convolutional Neural Network</a> . . . . .	9
<b>2.10</b>	Kiến trúc YOLO V3. Nguồn ảnh: <a href="#">Tomato Diseases and Pests Detection Based on Improved Yolo V3 Convolutional Neural Network</a> . . . . .	10
<b>2.11</b>	Phần code của backbone Darknet-53 . . . . .	10
<b>2.12</b>	Phần code của head Darknet-53 . . . . .	11
<b>2.13</b>	Example of Upsampling layer Nguồn ảnh: <a href="#">Improvement of Damage Segmentation Based on Pixel-Level Data Balance Using VGG-Unet</a> . . . . .	11
<b>2.14</b>	Ba kích thước lưới YOLO V3. Nguồn ảnh: <a href="#">Bài 25 - YOLO You Only Look Once</a> . . . . .	12
<b>2.15</b>	Kiến trúc của YOLO V3. Nguồn ảnh: <a href="#">What's new in YOLO v3?</a> . . . . .	12
<b>2.16</b>	Skip Connection Nguồn ảnh: <a href="#">Skip Connection</a> . . . . .	13
<b>2.17</b>	Output của YOLO V3. Nguồn ảnh <a href="#">Paperspace Blog</a> . . . . .	13
<b>2.18</b>	Anchor Box (nét đứt) và Bounding Box(nét liền màu xanh). Nguồn ảnh: <a href="#">YOLOv3: An Incremental Improvement</a> . . . . .	14
<b>2.19</b>	Loss function YOLO V3 . . . . .	14
<b>2.20</b>	Processing flow of YOLO V3 . . . . .	15
<b>2.21</b>	Processing flow of YOLO V3 . . . . .	15
<b>2.22</b>	Processing flow of YOLO V3 . . . . .	16
<b>2.23</b>	Processing flow of YOLO V3 . . . . .	16
<b>2.24</b>	Non - Max Suppression . . . . .	17
<b>2.25</b>	Intersection over Union . . . . .	17
<b>3.1</b>	Precision Recall Curve Graph . . . . .	19
<b>3.2</b>	Training lần 1 . . . . .	20
<b>3.3</b>	Đồ thị training lần 1 . . . . .	20
<b>3.4</b>	Training lần 2 . . . . .	21
<b>3.5</b>	Đồ thị training lần 2 . . . . .	22
<b>3.6</b>	Training lần 3 . . . . .	23
<b>3.7</b>	Đồ thị training lần 3 . . . . .	23

## CHƯƠNG 1

### GIỚI THIỆU ĐỀ TÀI

#### 1.1 Vấn đề bài toán đang giải quyết

Hiện nay tại các đô thị lớn như Thành phố Hồ Chí Minh, Hà Nội,... số lượng dân cư ngày càng tăng, kéo theo số lượng phương tiện giao thông tăng theo. Do đó hình thành nên những đoạn đường kẹt xe vào giờ cao điểm, gây ảnh hưởng đến môi trường và chất lượng cuộc sống.



Hình 1.1: Hình ảnh tình trạng kẹt xe diễn ra ở TP.Hồ Chí Minh

Nắm bắt được hiện trạng trên, em đã đưa ra ý tưởng về một mô hình AI, sử dụng Deep Learning để có thể nhận diện và đếm số lượng xe. Qua đó có thể thống kê được lưu lượng xe, dùng làm dữ liệu cho các nhà quy hoạch đường xá khắc phục tình trạng kẹt xe trên.

## 1.2 Ý tưởng giải quyết bài toán

Dể có thể giải quyết được bài toán trên, em quyết định sẽ sử dụng giải thuật YOLO v3 để thực hiện nhận diện và đếm các phương tiện giao thông đang lưu thông trên đường.

## CHƯƠNG 2

### GIẢI THUẬT YOLO V3

#### 2.1 Sơ lược về nhận diện vật thể - Object Detection

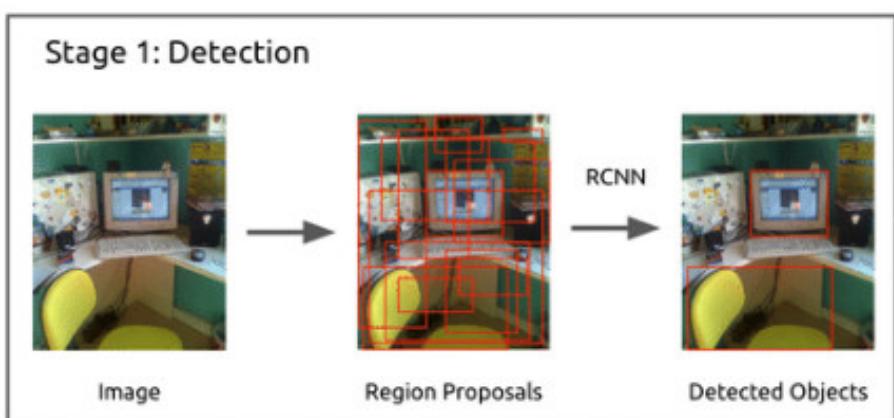
Object Detection (nhận biết vật thể) là bài toán về xác định vị trí và phân loại vật thể trong một bức hình.

So với Object Classification chỉ nhận diện một vật thể trên một bức hình, thì với cùng một bức hình như vậy, Object Detection có thể nhận diện nhiều vật thể. Hơn nữa, Object Detection còn xác định vị trí của vật thể thông qua bounding box bao quanh vật thể.

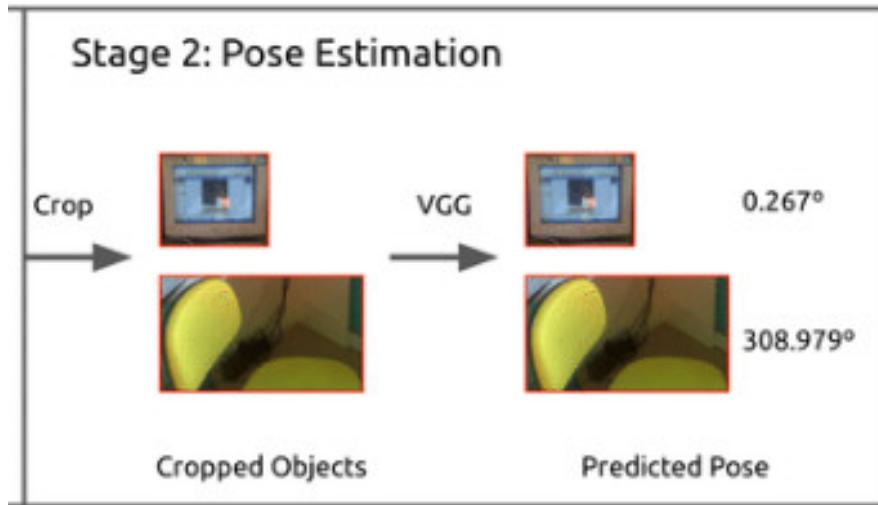
Thuật toán Object Detection chia làm hai nhóm chính, đó là:

- Họ các mô hình **Two Stage Detectors** dùng để giải quyết các bài toán về định vị và nhận diện vật thể.
- Họ các mô hình về **One Stage Detectors** dùng để nhận dạng đối tượng được thiết kế để nhận diện các vật thể real-time.

Họ các mô hình **Two Stage Detectors** thực hiện định vị, nhận diện vật thể của mình qua 2 giai đoạn. Giai đoạn một là xác định vị trí của vật thể, giai đoạn hai là phân loại và gán nhãn vật thể. Tiêu biểu cho mô hình này là RCNN, R-FCN, FPN,... Dưới đây là ví dụ cho việc sử dụng mô hình RCNN để nhận biết vật thể. Chúng ta hãy cùng xem hình ảnh để biết hai giai đoạn trên trong mô hình này được thực hiện như thế nào.

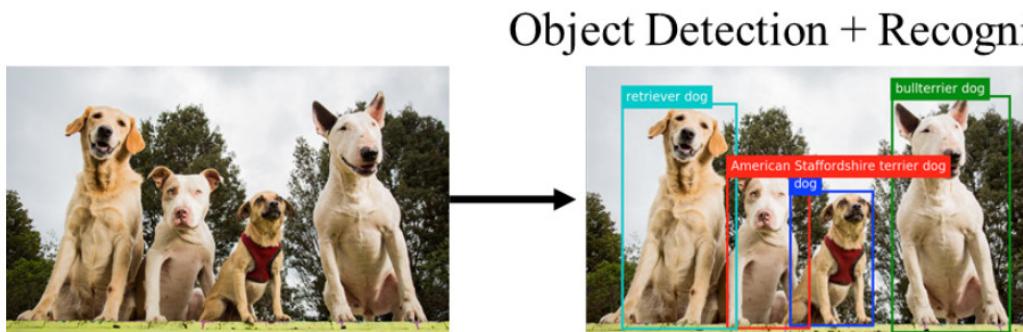


Hình 2.1: Giai đoạn 1



Hình 2.2: Giai đoạn 2

Họ các mô hình **One Stage Detectors** sẽ không phân ra hai giai đoạn rõ ràng, mà cùng lúc sẽ thực hiện cả việc tìm kiếm bounding box và classification cho một vật thể. Tiêu biểu cho mô hình này là các thuật toán YOLO, SSD,...



Hình 2.3: One stage detectors

## 2.2 Tổng quan về Convolutional Neural Network(CNN)

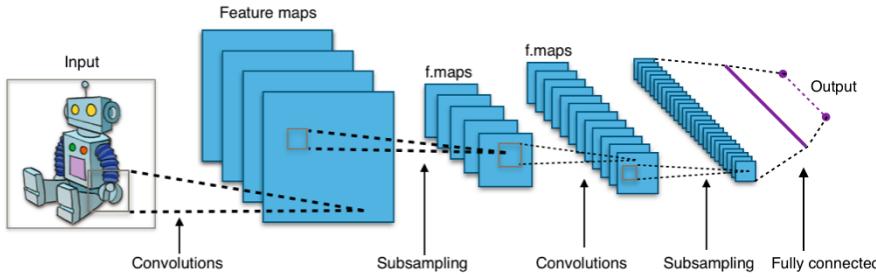
### 2.2.1 Giới thiệu về CNN

Convolutional Neural Network (CNN) là một trong những mô hình học sâu (deep learning). Nó giúp cho chúng ta xây dựng được những hệ thống thông minh với độ chính xác cao. CNN được dùng phổ biến trong các bài toán nhận diện vật thể. Trong bài toán này em áp dụng giải thuật CNN để xây dựng mô hình YOLO V3.

### 2.2.2 CNN hoạt động ra sao?

CNN khác với các mạng thần kinh khác về hiệu suất vượt trội của chúng vượt trội của chúng. Nó có 3 loại lớp chính:

- Convolution layer
- Pooling layer
- Fully-connected layer



Hình 2.4: Mô hình kiến trúc CNN điển hình.  
Nguồn ảnh: Wikipedia - Convolutional neural network

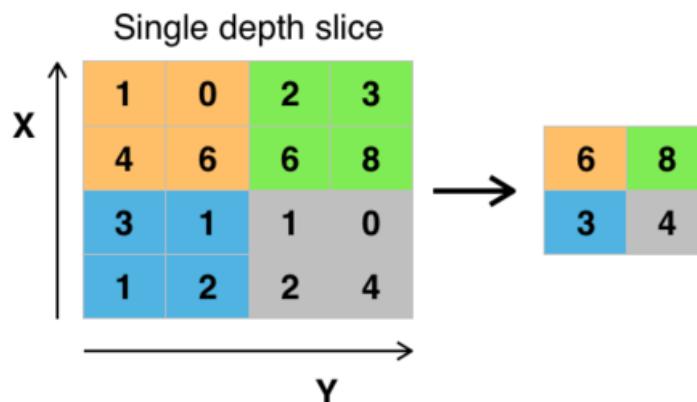
### Convolution Layer:

Convolution Layer là lớp cốt lõi trong việc xây dựng CNN và là nơi diễn là phần lớn việc tính toán. Nó yêu cầu các thành phần như: dữ liệu đầu vào, filter(bộ lọc) và feature map (bản đồ đặc trưng). Giả sử đầu vào là một hình ảnh màu được tạo từ ma trận pixel 3D. Có thể hiểu là dữ liệu đầu vào với ba kích thước: chiều dài, chiều rộng và chiều sâu. Với bộ nhận diện đặc trưng, được biết như hạt nhân hoặc filter, sẽ kiểm tra xem đặc trưng có tồn tại trong các trường của hình ảnh không. Quá trình này gọi là tích chập (convolution)

### Pooling layer:

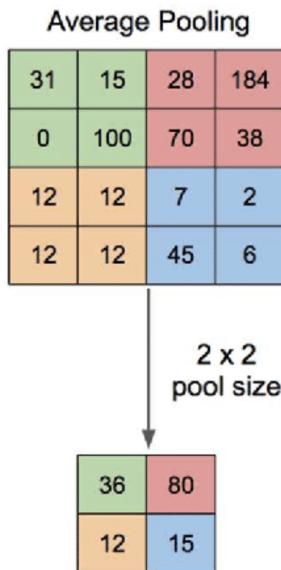
Pooling layer(lớp tổng hợp), hay còn gọi là downsampling, là lớp tiến hành giảm kích thước, giảm số lượng tham số của các trường đầu vào. Tương tự như lớp tích chập, lớp tổng hợp quét bộ lọc trên toàn bộ dữ liệu đầu vào, nhưng điểm khác biệt là trong bộ lọc này không có bất kỳ trọng số nào. Thay vào đó, hạt nhân áp dụng hàm tổng hợp cho các giá trị trong trường tiếp nhận, tạo nên các mảng đầu ra. Có hai loại tổng hợp chính:

- Max pooling: Khi bộ lọc di chuyển qua các trường, nó sẽ chọn pixel có giá trị lớn nhất để đưa đến mảng đầu ra. Bên cạnh đó, cách tiếp cận này có xu hướng được sử dụng thường xuyên hơn so với cách gộp trung bình.



Hình 2.5: Max pooling với bộ lọc 2x2 và bước nhảy 2. Nguồn ảnh: Wikipedia - Convolutional neural network

- Average pooling: Khi bộ lọc di chuyển qua đầu vào, nó sẽ tính toán giá trị trung bình trong các trường để đưa đến mảng đầu ra.

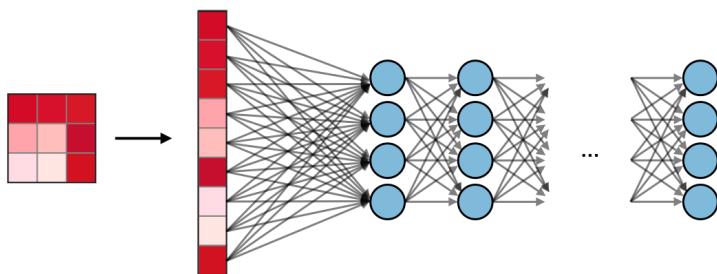


Hình 2.6: Average pooling với bộ lọc  $2 \times 2$  và bước nhảy 2. Nguồn ảnh: [Average Pooling](#)

Mặc dù thông qua lớp tổng hợp, nhiều thông tin có thể bị mất nhưng nó vẫn đem lại một số lợi ích cho CNN như giảm độ phức tạp, nâng cao hiệu quả, ...

#### Fully-connected layer:

Trong lớp kết nối đầy đủ, mỗi nút của lớp đầu ra đều kết nối với một nút của lớp trước đó. Lớp này thực hiện nhiệm vụ phân loại dựa trên các đặc trưng được trích xuất thông qua các lớp và các filter trước đó.



Hình 2.7: Fully-connection layer. Nguồn ảnh: [Mạng neural tích chập cheatsheet](#)

## 2.3 YOLO V3 là gì?

Sau khi tìm hiểu sơ lược về Object Detection. Em sẽ đề cập chi tiết hơn về giải thuật YOLO, và cụ thể là phiên bản YOLO V3.

YOLO V3 (You Only Look Once, Version 3) là thuật toán nhận diện vật thể thời gian thực (real-time object detection) với khả năng xác định các đối tượng cụ thể trong các hình ảnh, video hoặc phương tiện truyền phát trực tuyến.

Giải thuật học máy YOLO sử dụng mạng thần kinh tích chập (Convolutional neural network) để phát hiện, nhận dạng và phân loại đối tượng. YOLO được tạo ra từ việc kết hợp giữa các convolutional layers và fully-connected layers. Trong đó các convolutional layers sẽ trích xuất ra các feature của ảnh, còn fully-connected layers sẽ dự đoán ra xác suất đó và tọa độ của đối tượng.

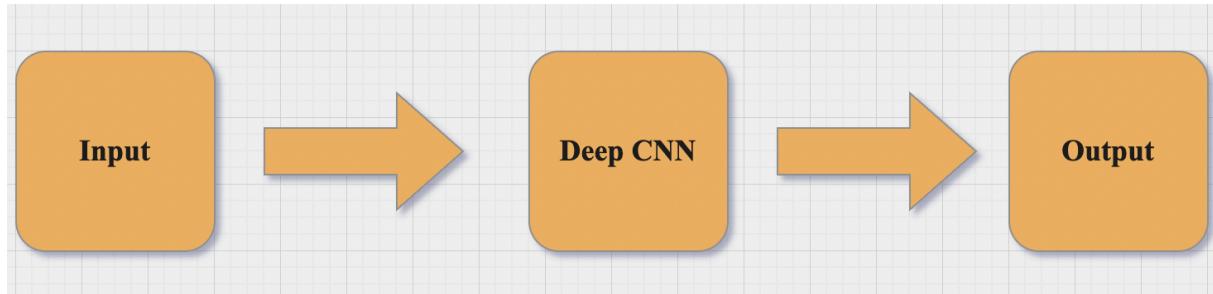
Các phiên bản YOLO từ 1 cho đến 3 đều được tạo ra bởi Joseph Redmon. Ông đã phát triển ba phiên bản YOLO, với phiên bản thứ ba là YOLO v3, được viết bằng kiến trúc Darknet. YOLO v3 cho kết quả chính xác hơn hai phiên bản trước đó. Glenn Jocher đã sử dụng phiên bản YOLOv3 trong PyTorch với một số thay đổi nhỏ và đặt tên nó là YOLO v5. Kiến trúc YOLO v5 sau đó đã được điều chỉnh để phát triển YOLO v8, nó vẫn đang được tích cực nghiên cứu và phát triển.

YOLO V3 là phiên bản YOLO mà em chọn để hiện thực mô hình của mình.

## 2.4 Cách hoạt động của YOLO V3

YOLO V3 hay YOLO hoạt động theo cơ chế detect vật thể trên từng vùng ảnh. Ảnh được chia thành  $S * S$  ô, hay còn gọi là cell. Thực ra là chia trong trí tưởng tượng, không phải cắt ảnh ra hay thực hiện bất kì bước image - processing nào. Bản chất của việc chia ảnh là việc chia output, target thành matrix A có kích thước  $S * S$ . Nếu trong ảnh, tâm của vật thể nằm trong cell thứ (i, j) thì output tương ứng nó sẽ nằm trong  $A[i, j]$ .

YOLO V3 hoạt động theo mô hình sau đây:



Hình 2.8: Mô hình thực hiện YOLO V3

Từ mô hình, ta thấy Input của YOLO V3, sau khi đi qua các mạng CNN, chúng ta sẽ nhận được output.

### 2.4.1 Input

Input của YOLO V3, bao gồm hình ảnh, video,...

### 2.4.2 Kiến trúc mạng CNN của YOLO V3

Trong YOLO V3, xương sống của mạng (backbone network) là DarkNet-53 với 53 lớp chập. Kiến trúc của DarkNet - 53 được thể hiện như trong hình dưới đây.

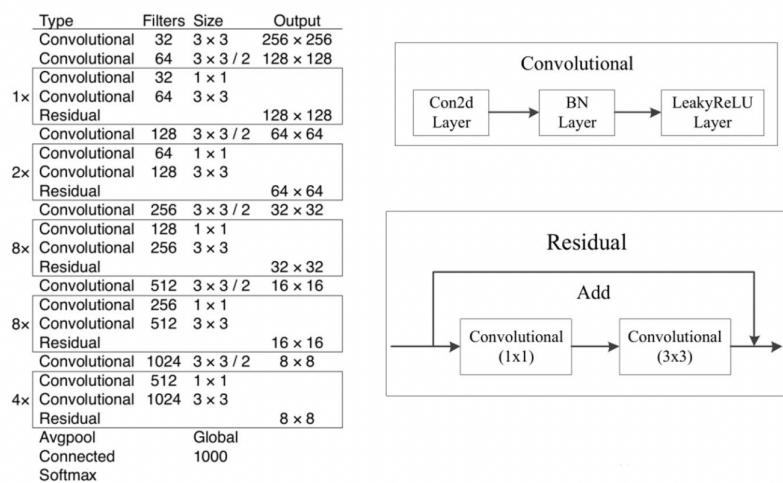


FIGURE 1 | Darknet-53 network structure.

Hình 2.9: DarkNet - 53.

Nguồn ảnh: Tomato Diseases and Pests Detection Based on Improved Yolo V3 Convolutional Neural Network

Kiến trúc của YOLO V3 được thể hiện trong hình sau:

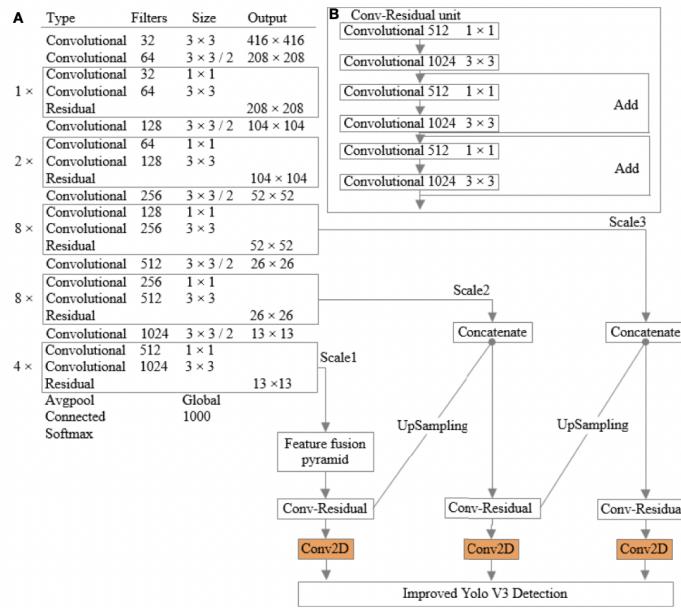


FIGURE 2 | The improved Yolo V3 network structure (A) network structure; (B) conv-residual unit.

Hình 2.10: Kiến trúc YOLO V3.

Nguồn ảnh: Tomato Diseases and Pests Detection Based on Improved Yolo V3 Convolutional Neural Network

```
# darknet53 backbone
backbone:
    # [from, number, module, args]
    [[-1, 1, Conv, [32, 3, 1]],  # 0
     [-1, 1, Conv, [64, 3, 2]],  # 1-P1/2
     [-1, 1, Bottleneck, [64]],
     [-1, 1, Conv, [128, 3, 2]],  # 3-P2/4
     [-1, 2, Bottleneck, [128]],
     [-1, 1, Conv, [256, 3, 2]],  # 5-P3/8
     [-1, 8, Bottleneck, [256]],
     [-1, 1, Conv, [512, 3, 2]],  # 7-P4/16
     [-1, 8, Bottleneck, [512]],
     [-1, 1, Conv, [1024, 3, 2]], # 9-P5/32
     [-1, 4, Bottleneck, [1024]], # 10
    ]
```

Hình 2.11: Phần code của backbone Darknet-53

```

# YOLOv3 head
head:
[[ -1, 1, Bottleneck, [1024, False]],
 [-1, 1, Conv, [512, 1, 1]],
 [-1, 1, Conv, [1024, 3, 1]],
 [-1, 1, Conv, [512, 1, 1]],
 [-1, 1, Conv, [1024, 3, 1]], # 15 (P5/32-large)

[-2, 1, Conv, [256, 1, 1]],
[-1, 1, nn.Upsample, [None, 2, 'nearest']],
[[-1, 8], 1, Concat, [1]], # cat backbone P4
[-1, 1, Bottleneck, [512, False]],
[-1, 1, Bottleneck, [512, False]],
[-1, 1, Conv, [256, 1, 1]],
[-1, 1, Conv, [512, 3, 1]], # 22 (P4/16-medium)

[-2, 1, Conv, [128, 1, 1]],
[-1, 1, nn.Upsample, [None, 2, 'nearest']],
[[-1, 6], 1, Concat, [1]], # cat backbone P3
[-1, 1, Bottleneck, [256, False]],
[-1, 2, Bottleneck, [256, False]], # 27 (P3/8-small)

[[27, 22, 15], 1, Detect, [nc, anchors]], # Detect(P3, P4, P5)
]

```

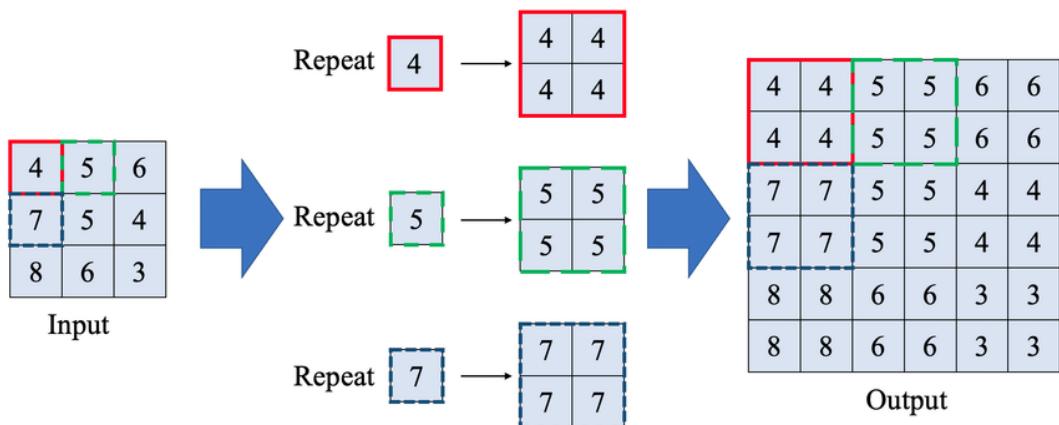
Hình 2.12: Phần code của head Darknet-53

Từ hình ảnh về kiến trúc của YOLO V3 và các phần code, ta thấy, đầu tiên khi nhận input là một bức ảnh, hình ảnh này sẽ đi qua 53 lớp tích chập (Convolutional Layer) của DarkNet - 53 và các skip connection và thu được kết quả ảnh. Và sau đó ảnh được nhận diện qua 3 đầu dự đoán (head) bằng cách áp dụng convolutional set và convolutional 2d 1x1 trên bản đồ đặc trưng, ở ba kích thước khác nhau bằng cách thay ảnh đầu vào chia cho 32, 16 và 8 và ở ba vị trí khác nhau (P5, P4, P3 như trong hình 2.12) tương ứng trong mạng, thu được kết quả cuối cùng.

Trong quá trình phân tích qua YOLOv3 head, bản đồ đặc trưng phải đi qua 3 lớp convolutional  $1 \times 1$ , upsampling và concatenate.

Trong đó:

Lớp upsampling là lớp đơn giản không có trọng số, nó sẽ nhân đổi chiều của ảnh đầu vào.



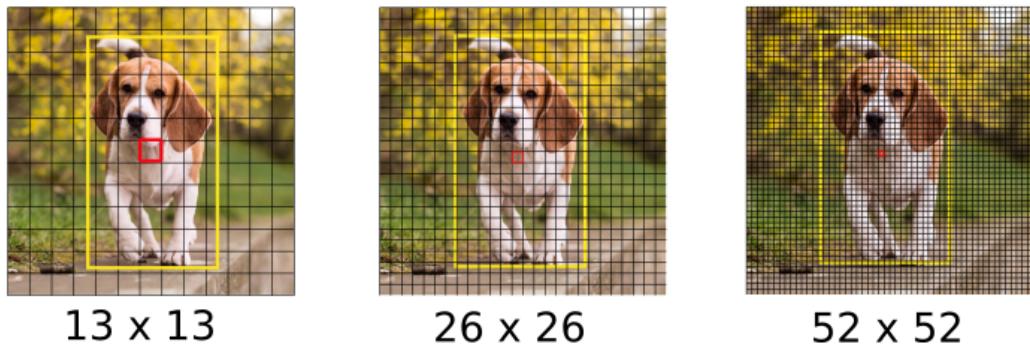
Hình 2.13: Example of Upsampling layer

Nguồn ảnh: Improvement of Damage Segmentation Based on Pixel-Level Data Balance Using VGG-Unet

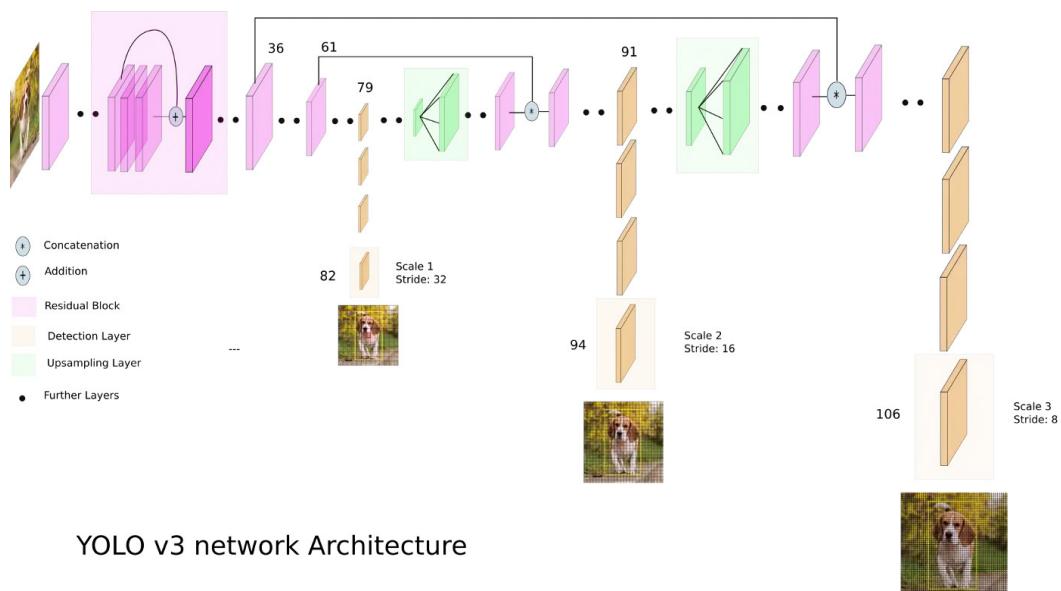
Lớp concatenate: Kết hợp đầu ra hiện tại với đầu ra của skip connection cụ thể trước đó. Có thể thấy trên hình, đầu ra của lớp skip connection 36 kết hợp với đầu ra của lớp upsampling 97 và đầu ra của lớp

skip connection 61 kết hợp với đầu ra của lớp upsampling 85. Từ đó thu được đầu ra từ lớp concatenate 98 và 86.

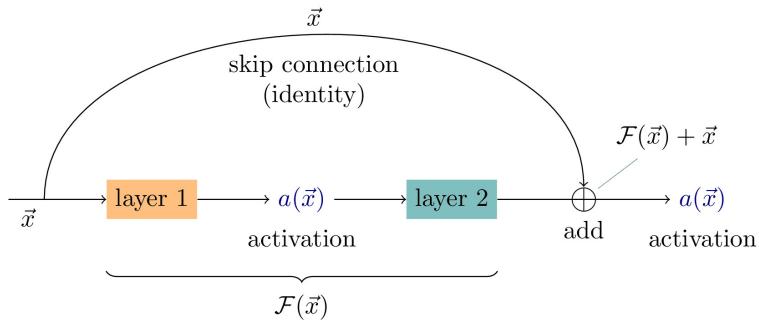
Từ đó, chúng ta sẽ dự đoán bounding box dựa trên 3 kích thước lưới khác nhau. Với các vật thể lớn thì sẽ dễ dàng phát hiện trên các lưới thô và ngược lại đối với các vật thể nhỏ hơn trên lưới mịn. Trong YOLOv3, kích thước lưới thường dùng là [13, 26, 52] cho hình ảnh có kích thước  $416 \times 416$ . Còn nếu sử dụng ảnh có kích thước khác thì kích thước lưới đầu tiên sẽ là kích thước ảnh chia cho 32 và những cái còn lại sẽ là gấp đôi so với cái trước đó.



Hình 2.14: Ba kích thước lưới YOLO V3.  
Nguồn ảnh: Bài 25 - YOLO You Only Look Once



Hình 2.15: Kiến trúc của YOLO V3.  
Nguồn ảnh: What's new in YOLO v3?



Hình 2.16: Skip Connection  
Nguồn ảnh: Skip Connection

### 2.4.3 Output

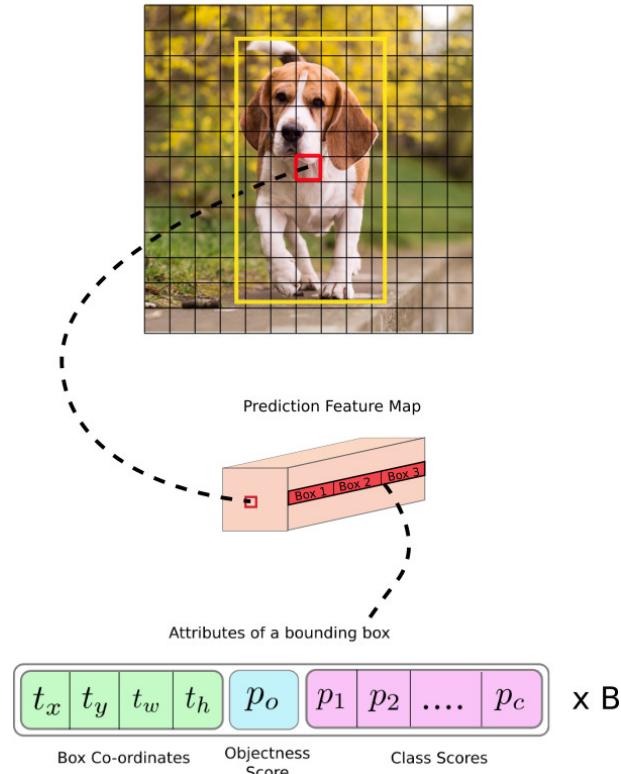
Output của YOLO V3 là một matrix A có kích thước:

$$S * S * B * (4 + 1 + C)$$

Trong đó:

$S * S$ : chính là số ô mà ảnh được chia ra. Như đã đề cập ở trên, đây là việc chia trong trí tưởng tượng.  
 $B$ : là số bounding box trên mỗi ô. Mỗi bounding box có  $5 + C$  thuộc tính, bao gồm:  $t_x, t_y, t_w, t_h, p_1, p_2, p_3, p_4, \dots$  được thể hiện trong ảnh sau đây:

Image Grid. The Red Grid is responsible for detecting the dog



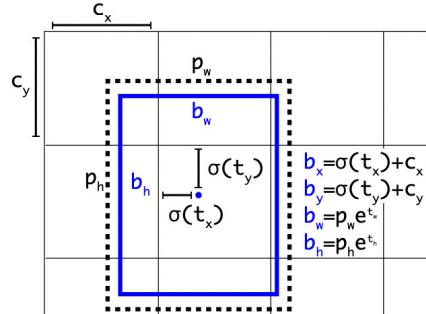
Hình 2.17: Output của YOLO V3. Nguồn ảnh Paperspace Blog

Trong output được trả về, các tham số Box Coordinates  $t_x, t_y, t_w, t_h$  không phải giá trị thực của bbox, mà chỉ là giá trị offset của bbox so với 1 anchor box cho trước. Anchor box này có kích thước ( $p_w, p_h$ ) được định nghĩa sẵn. Từ  $t_x, t_y, t_w, t_h$  ta cần bước biến đổi để tính ra giá trị tọa độ thật của bounding

box  $(b_x, b_y, b_h, b_w)$ , lần lượt là tọa độ tâm x, y, chiều rộng và chiều cao của bounding box, dựa trên  $(p_w, p_h)$  theo công thức dưới đây, với các hàm  $\sigma(x)$  là các hàm sigmoid:

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

Ta có  $c_x, c_y$  là tọa độ trên cùng bên trái của ô.



Hình 2.18: Anchor Box (nét đứt) và Bounding Box(nét liền màu xanh).  
Nguồn ảnh: YOLOv3: An Incremental Improvement

Ở đây chúng ta dự đoán tọa trung tâm bằng hàm sigmoid. Điều này bắt buộc kết quả đầu ra nằm trong khoảng từ 0 đến 1. Chúng ta xem xét ví dụ về hình ảnh con chó ở bên trên, bounding box được thể hiện trên ảnh. Nếu tọa độ dự đoán của điểm trung tâm bounding box là (0.6, 0.7) thì điều này có nghĩa là điểm trung tâm nằm ở vị trí (6.6, 6.7) trên feature map, vì tọa độ trên cùng bên trái của ô màu đỏ là (6, 6).

Về các tham số còn lại, ta có Objectness Score po là xác suất tại bounding box đó có chứa object hay không. Class score:  $p_1, p_2, p_3, \dots, p_c$  đại diện cho xác suất của object lược về một class cụ thể trong C class (chó, mèo, người, xe đạp,...).

#### 2.4.4 Loss function

$$\begin{aligned} \lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^B \mathbb{1}_{ij}^{\text{obj}} &\left[ (x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 \right] \\ + \lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^B \mathbb{1}_{ij}^{\text{obj}} &\left[ (\sqrt{w_i} - \sqrt{\hat{w}_i})^2 + (\sqrt{h_i} - \sqrt{\hat{h}_i})^2 \right] \\ + \sum_{i=0}^{S^2} \sum_{j=0}^B \mathbb{1}_{ij}^{\text{obj}} &[\hat{C}_i \log(C_i) + (1 - \hat{C}_i) \log(1 - C_i)] \\ + \lambda_{\text{noobj}} \sum_{i=0}^{S^2} \sum_{j=0}^B \mathbb{1}_{ij}^{\text{noobj}} &[\hat{C}_i \log(C_i) + (1 - \hat{C}_i) \log(1 - C_i)] \\ + \sum_{i=0}^{S^2} \mathbb{1}_i^{\text{obj}} \sum_{c \in \text{classes}} &[\hat{p}_i(c) \log(p_i(c)) + (1 - \hat{p}_i(c)) \log(1 - p_i(c))] \end{aligned}$$

Hình 2.19: Loss function YOLO V3

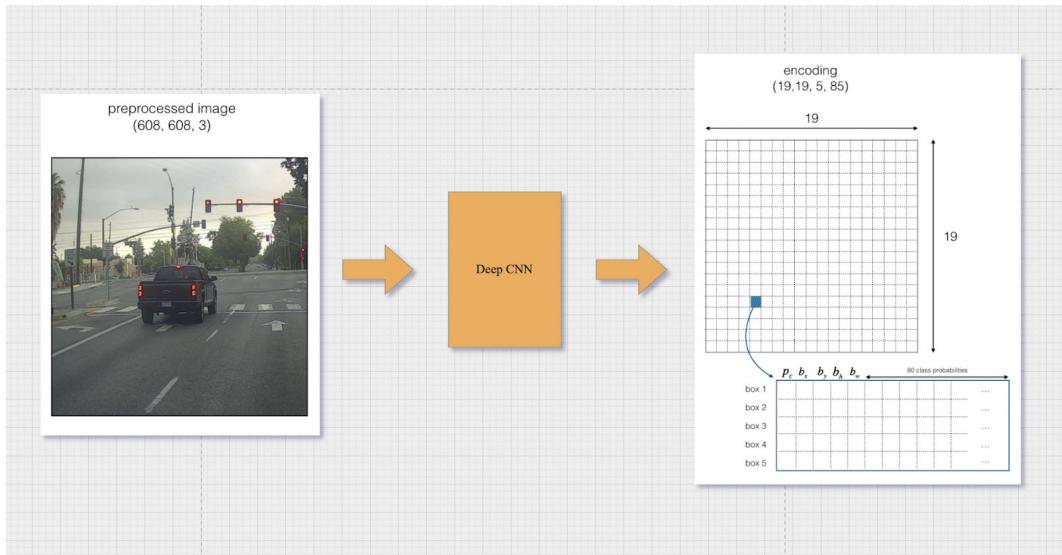
Giải thích:

- $\mathbb{1}_{ij}^{\text{obj}}$  là hàm tồn tại trả về 1 nếu có vật thể hoặc 0 nếu không có vật thể
- x,y,w,h lần lượt là tọa độ tâm x, y, độ rộng và độ cao của bounding box

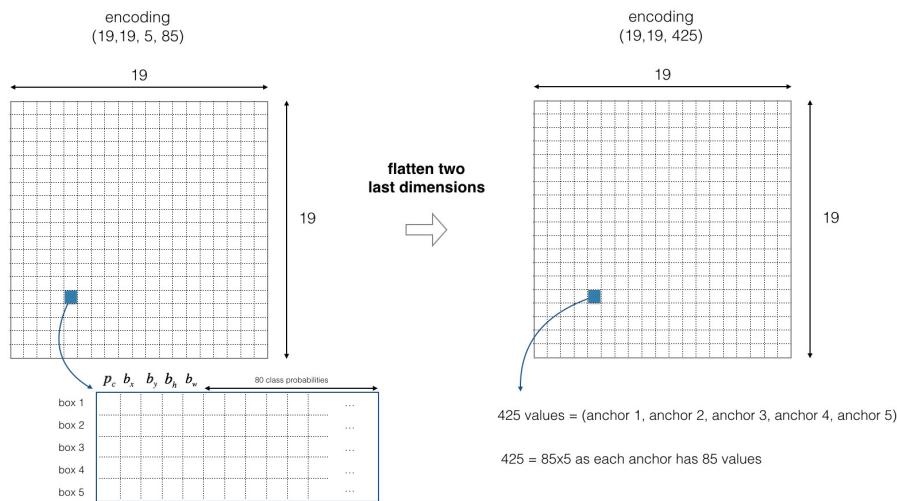
- $p(c)$  là xác suất của class  $c$  trong bounding box
- $C$  là độ tin cậy có vật thể trong bounding box
- các biến có mũ trên đầu là giá trị thực tế, giá trị thật còn các biến không có mũ là các giá trị dự đoán
- các hằng số  $\lambda$  là các lượng giá với  $\lambda_{coord}$  lớn hơn  $\lambda_{noobj}$

#### 2.4.5 Luồng xử lý của YOLO V3

Để có thể dễ hiểu hơn về luồng thực thi của YOLO V3. Em sẽ lấy ví dụ minh họa như hình sau:

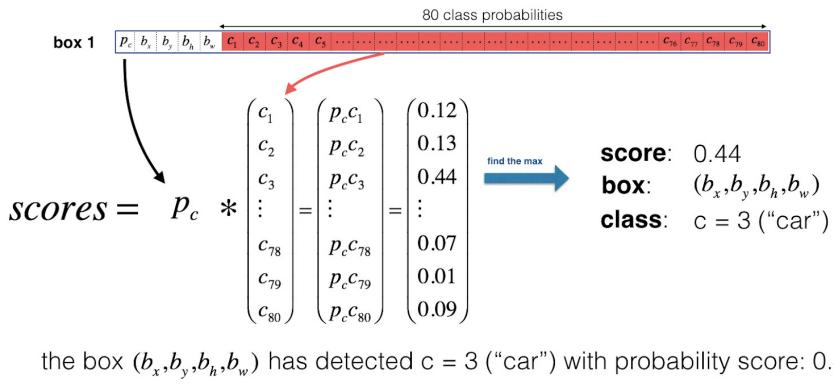


Hình 2.20: Processing flow of YOLO V3



Hình 2.21: Processing flow of YOLO V3

Trong ví dụ trên, chúng có một ảnh đầu vào kích thước  $608 * 608 * 3$ , sau khi cho qua mạng CNN của YOLO V3, chúng ta nhận được một tensor có shape  $(19, 19, 5, 85)$ , 19 có nghĩa là hình ảnh của chúng ta được chia thành  $19 * 19$  ô, 5 là số bounding box mà mỗi ô trong  $19 * 19$  ô dự đoán ra, 85 ở đây bao gồm  $b_x, b_y, b_w, b_h, p_o$  (trong hình 2.20 là  $p_c$ ) và xác suất cho 80 classes, từ  $c_1, c_2, \dots, c_{80}$ .



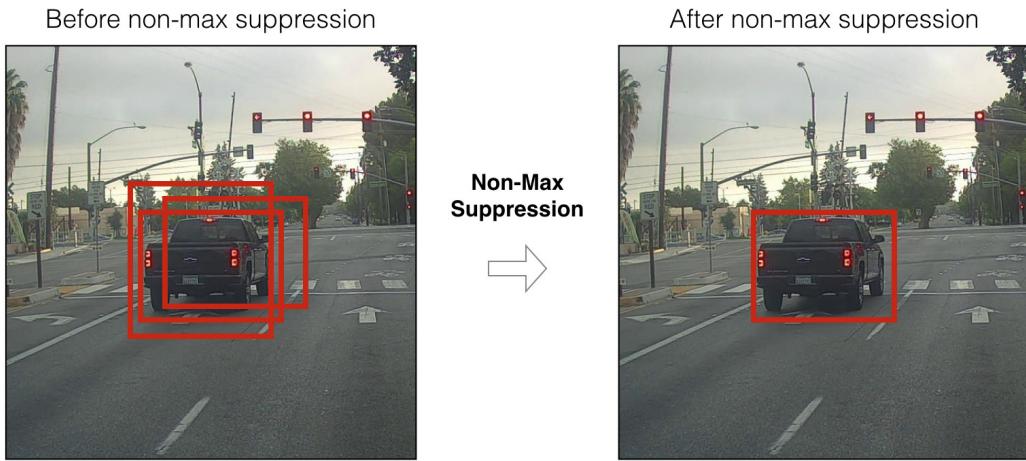
Hình 2.22: Processing flow of YOLO V3

Từ mỗi box trong một ô, chúng ta sẽ đi tìm ra box này sẽ dự đoán ra class nào thông qua phép nhân  $p_c$  với một tensor chứa xác suất cho mỗi class. Sau đó chúng ta chọn ra phần tử có giá trị lớn nhất trong tensor kết quả. Trong ví dụ trên, ta thấy rằng box 1 dự đoán ra object là "car". Sau đó chúng ta lại làm tương tự với các box khác trong cùng 1 ô với nhau, và cuối cùng chọn ra class nào có score lớn nhất làm class mà ô này dự đoán ra. Chúng ta có thể hình dung kết quả dự đoán của các ô qua hình sau, bằng cách tô màu các ô theo class mà nó dự đoán.



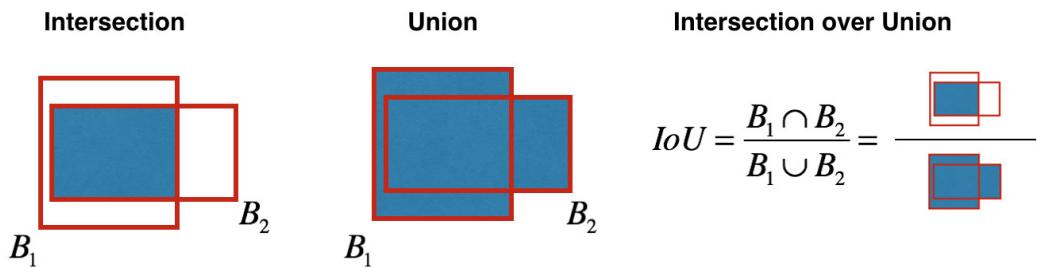
Hình 2.23: Processing flow of YOLO V3

Sau khi có được các giá trị classes scores, chúng ta sẽ tiến hành lọc giá trị này bằng một ngưỡng giá trị, gọi là thresholding. Giả sử giá trị này bằng 0.5, thì classes scores của ô nào có giá trị nhỏ hơn 0.5 thì ta sẽ loại bỏ. Khi loại bỏ xong các giá trị classes scores không cao, chúng ta phải quyết thêm một vấn đề nữa, đó chính là có nhiều bounding box dự đoán cùng một class, khiến chúng nằm chồng chéo lên nhau. Bộ lọc thứ hai gọi là Non - Max Suppression sẽ giúp chúng ta giải quyết điều đó.



Hình 2.24: Non - Max Suppression

Non - Max Suppression sử dụng hàm IOU (Intersection over Union). Hàm IOU được thể hiện trong hình sau:



Hình 2.25: Intersection over Union

Để hiện thực Non - Max Suppression, ta làm theo các bước như sau: Đầu tiên chúng ta chọn ra box có class score cao nhất, sau đó ta lần lượt tiến hành tính IOU giữa box đó với các box khác. Tiến hành xóa box có class score nhỏ hơn nếu kết quả hàm IOU lớn hơn một giá trị ngưỡng iou\_threshold. Ta cứ xóa như vậy cho đến khi chỉ còn một bounding box duy nhất. Và cuối cùng chúng ta thu được output của YOLO V3.

## 2.5 Điểm mới của YOLO V3 so với các phiên bản trước đó

Trước khi YOLO có được phiên bản 3 thì hãy điểm lại những đặc điểm nổi bật mà các phiên bản sau đã khắc phục các phiên bản trước như thế nào:

Ban đầu YOLO V1 vẫn có nhiều nhược điểm cần khắc phục:

- Độ chính xác vẫn còn kém so với các Region-based detector.
- Do thiết kế output của YOLO V1, ta chỉ có thể dự đoán tối đa một object trong 1 cell (vì chỉ có một class\_distribution cho mỗi cell). Thường gặp rắc rối khi nhận diện 1 nhóm vật thể nhỏ trong cell như 1 đàn chim.
- Điều đó có nghĩa rằng nếu ta chọn  $S \times S = 7 \times 7$ , số lượng object tối đa chỉ bằng 49. Với những trường hợp nhiều vật cùng nằm trong 1 cell, YOLO sẽ kém hiệu quả.
- $(x,y,w,h)$  được predict ra trực tiếp  $\rightarrow$  giá trị tự do. Trong khi đó trong hầu hết các dataset, các bounding box có kích cỡ không quá đa dạng mà tuân theo những tỷ lệ nhất định.

YOLO V2 ra đời nhằm cải thiện những vấn đề này. Các cải tiến của YOLO V2 so với YOLO V1 bao gồm:

- ImageNet 1000-class sang Darknet-19
- Input image có kích thước lớn hơn bản v1 từ 224x224 thành 448x448
- Tận dụng ý tưởng anchor boxes convolutional từ FasterRCNN
- High Resolution Classifier
- Thêm các Batch Normalization layer vào model
- Bỏ đi các fully-connected ở cuối model, giúp tăng tốc độ tính toán, kích thước ảnh input cũng động hơn
- Thay vì có chỉ một class\_distribution chung cho cả một cell, mỗi bounding box đều có class\_distribution của riêng nó. Như vậy, một cell có thể đồng thời có nhiều boundbox cho các object khác nhau.

Nhược điểm YOLO V2

- Vấn đề predict vị trí trực tiếp. Khi sử dụng anchor boxes thì model không ổn định.
- Input image có giới hạn

Đến với YOLO V3, đã có sự thay đổi so với phiên bản trước V2 bao gồm:

- Darknet-19 sang Darknet-53 khiến số lượng layer lên đến 106
- Với số lượng convolution layer lớn giúp cho độ chính xác tăng lên nhưng tốc độ giảm xuống
- Cũng nhờ vậy YOLO V3 nhận diện các vật thể nhỏ tốt hơn
- Loss function của YOLO V3 thay đổi so với YOLO V2
- Cụ thể là độ tin cậy của object và dự đoán class trong YOLO v3 được dự đoán thông qua hồi quy logistic (cross-entropy).

## CHƯƠNG 3

### HIỆN THỰC MÔ HÌNH

#### 3.1 Dữ liệu huấn luyện mô hình

- Dữ liệu dùng để training là dữ liệu có sẵn trên [Roboflow](#)
- Dữ liệu gồm có 930 bức ảnh đã gán nhãn sẵn với hai nhãn là cars và truck
- Đường dẫn đến dataset [nhấn vào đây](#)
- Đường dẫn đến Drive tổng hợp đồ án [nhấn vào đây](#)

#### 3.2 Một số thông số của Model Summary

##### 3.2.1 Precision (P)

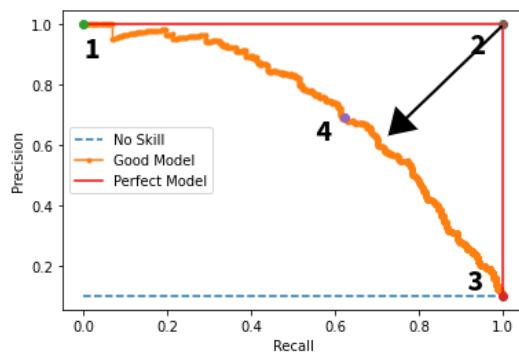
Precision đại diện cho độ tin cậy của Model. Cách tính của Precision là lấy tổng số true positive chia cho tổng số positive. Có nghĩa là cho biết phần trăm dự đoán chính xác của model.

##### 3.2.2 Recall (R)

Recall đại diện cho khả năng phát hiện chính xác các true positive của model. Ví dụ ta có n true positive, model dự đoán ra được m true positive thì Recall =  $m/n$ .

##### 3.2.3 Precision Recall Curve

Các giá trị dự đoán nào lớn hơn ngưỡng Thresh (IOU>Thresh) thì được gọi là true positive, nhỏ hơn thì gọi là false positive. Từ điều này ta tăng giảm Thresh từ 0 đến 1 với bước nhảy nhất định thì sẽ vẽ được một đồ thị giữa Recall và Precision. Đồ thị này gọi là Precision Recall Curve.



Hình 3.1: Precision Recall Curve Graph

### 3.2.4 Average Precision (AP)

AP là vùng diện tích nằm phía dưới đường cong Precision Recall Curve, Vùng diện tích này càng lớn thì đối với các giá trị Thresh khác nhau thì Recall và Precision cũng cho ra giá trị gần 1, tức là model tốt. Ngược lại AP nhỏ thì model không tốt.

### 3.2.5 mAP

mAP là trung bình tất cả các giá trị AP trong quá trình training. mAP 0.5 là trung bình tất cả các giá trị AP đối với ngưỡng thresh là 0.5. mAP 0.5:0.95 là trung bình tất cả các giá trị AP đối với ngưỡng thresh từ 0.5 đến 0.95.

## 3.3 Tổng hợp những lần training

### 3.3.1 Lần 1

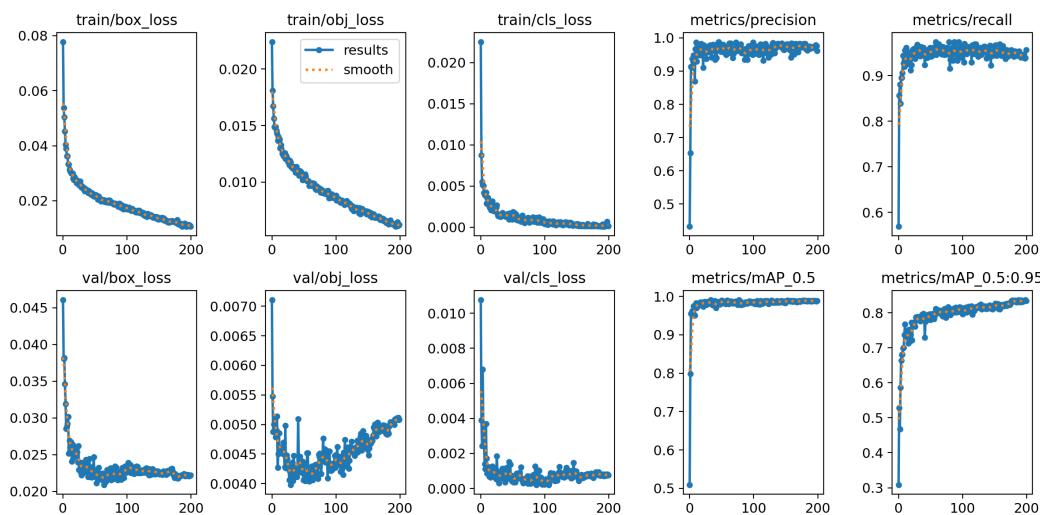
- Các thông số training

- img: 416
- batch: 16
- epochs: 200
- weights: yolov5s.pt

- Kết quả training

```
COMET INFO: Metrics [count] (min, max):
COMET INFO: loss [2020] : (0.16670012474060059, 2.713331298828125)
COMET INFO: metrics/mAP_0.5 [400] : (0.5092552019679712, 0.9909945566151995)
COMET INFO: metrics/mAP_0.5:0.95 [400] : (0.30865620047789954, 0.8366078825292096)
COMET INFO: metrics/precision [400] : (0.4325039708930061, 0.988493770624294)
COMET INFO: metrics/recall [400] : (0.5689176617579366, 0.9744340779062405)
COMET INFO: train/box_loss [400] : (0.010651599615812302, 0.07775428891181946)
COMET INFO: train/cls_loss [400] : (8.182025339920074e-05, 0.02253958024084568)
COMET INFO: train/obj_loss [400] : (0.006081387400627136, 0.02240639738738537)
COMET INFO: val/box_loss [400] : (0.020916571840643883, 0.0460679829120636)
COMET INFO: val/cls_loss [400] : (0.0002125235455904156, 0.010760975070297718)
COMET INFO: val/obj_loss [400] : (0.003981778863817453, 0.00710643082857132)
COMET INFO: x/lr0 [400] : (0.00019900000000000001, 0.07027272727272728)
COMET INFO: x/lr1 [400] : (0.00019900000000000001, 0.009870996969696968)
COMET INFO: x/lr2 [400] : (0.00019900000000000001, 0.009870996969696968)
```

Hình 3.2: Training lần 1



Hình 3.3: Đồ thị training lần 1

- **Dánh giá kết quả training**

- **Loss:**

**loss [2020]:** Dao động từ 0.1667 đến 2.7133, có vẻ ổn định nhưng còn khá cao.

**train/box\_loss, train/cls\_loss, train/obj\_loss:** Các giá trị này nằm trong khoảng chấp nhận được và giảm dần theo thời gian, cho thấy việc học đang diễn ra tốt.

- **Metrics mAP (Mean Average Precision):**

**metrics/mAP\_0.5:** Mean Average Precision với overlap threshold 0.5 nằm trong khoảng từ 0.5092 đến 0.9910. Giá trị cao của mAP\_0.5 là một dấu hiệu tích cực.

**metrics/mAP\_0.5:0.95:** Một metric khác với các giá trị từ 0.3087 đến 0.8366. Càng cao thì mô hình càng tốt.

- **Metrics Precision và Recall:**

**metrics/precision:** Nằm trong khoảng từ 0.4325 đến 0.9884.

**metrics/recall:** Nằm trong khoảng từ 0.5689 đến 0.9744.

Cả hai metrics này đều có giá trị khá cao, điều này có nghĩa là mô hình có khả năng đảm bảo độ chính xác và độ phủ (recall) tốt.

- **Learning Rates (lr):**

**x/lr0, x/lr1, x/lr2:** Tất cả các learning rates đều có giá trị khá thấp, điều này có thể giúp mô hình học tốt hơn.

- **Losses trên tập validation (val):**

**val/box\_loss, val/cls\_loss, val/obj\_loss:** Các giá trị loss trên tập validation đều khá thấp, cho thấy mô hình không bị overfitting nhiều trên tập huấn luyện.

### 3.3.2 Lần 2

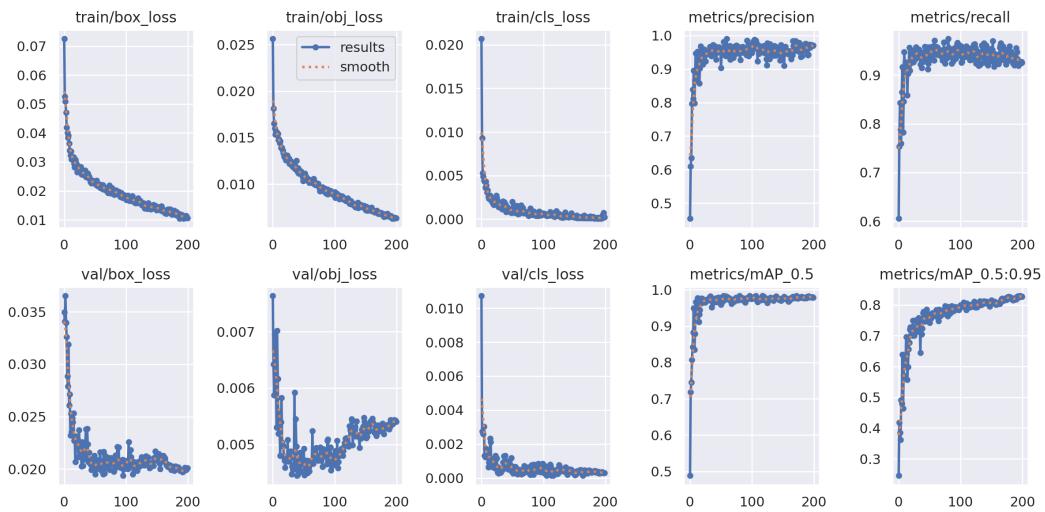
- **Các thông số training**

- img: 640
- batch: 16
- epochs: 200
- weights: yolov5s.pt

- **Kết quả training**

<b>COMET INFO:</b>	<b>Metrics [count] (min, max):</b>
<b>COMET INFO:</b>	<b>loss [2020]</b> : (0.15217134356498718, 3.001668930053711)
<b>COMET INFO:</b>	<b>metrics/mAP_0.5 [400]</b> : (0.48851294901238074, 0.9832474918700124)
<b>COMET INFO:</b>	<b>metrics/mAP_0.5:0.95 [400]</b> : (0.24542060767748067, 0.829536881313977)
<b>COMET INFO:</b>	<b>metrics/precision [400]</b> : (0.453650316572341, 0.9906342772602315)
<b>COMET INFO:</b>	<b>metrics/recall [400]</b> : (0.6053671103477887, 0.9745823101015639)
<b>COMET INFO:</b>	<b>train/box_loss [400]</b> : (0.010429446585476398, 0.07254188507795334)
<b>COMET INFO:</b>	<b>train/cls_loss [400]</b> : (6.210527499206364e-05, 0.020697979256510735)
<b>COMET INFO:</b>	<b>train/obj_loss [400]</b> : (0.006289147771894932, 0.025628600269556046)
<b>COMET INFO:</b>	<b>val/box_loss [400]</b> : (0.019365563988685608, 0.036526869982481)
<b>COMET INFO:</b>	<b>val/cls_loss [400]</b> : (0.00014517705130856484, 0.010728581808507442)
<b>COMET INFO:</b>	<b>val/obj_loss [400]</b> : (0.004446288105100393, 0.007634838577359915)
<b>COMET INFO:</b>	<b>x/lr0 [400]</b> : (0.00019900000000000001, 0.07027272727272728)
<b>COMET INFO:</b>	<b>x/lr1 [400]</b> : (0.00019900000000000001, 0.00987099696969696968)
<b>COMET INFO:</b>	<b>x/lr2 [400]</b> : (0.00019900000000000001, 0.009870996969696968)

Hình 3.4: Training lần 2



Hình 3.5: Đồ thị training lần 2

- **Đánh giá kết quả training**

- **Loss:**

**loss [2020]:** Dao động từ 0.1522 đến 3.0017, có vẻ ổn định và thấp hơn so với một số giá trị trong phạm vi 0.1667 đến 2.7133 của lần training trước.

**train/box\_loss, train/cls\_loss, train/obj\_loss:** Các giá trị loss đều khá thấp và giảm dần theo thời gian, cho thấy quá trình học diễn ra tốt.

- **Metrics mAP (Mean Average Precision):**

**metrics/mAP\_0.5:** Mean Average Precision với overlap threshold 0.5 nằm trong khoảng từ 0.4885 đến 0.9832, giảm so với lần trước nhưng vẫn ổn định và cao.

**metrics/mAP\_0.5:0.95:** Giảm từ 0.2454 đến 0.8295, nhưng vẫn ở mức khá cao.

- **Metrics Precision và Recall:**

**metrics/precision:** Nằm trong khoảng từ 0.4537 đến 0.9906.

**metrics/recall:** Nằm trong khoảng từ 0.6054 đến 0.9746.

Cả hai metrics này vẫn rất cao, thể hiện khả năng chính xác và độ phủ của mô hình.

- **Learning Rates (lr):**

**x/lr0, x/lr1, x/lr2:** Tất cả các learning rates đều có giá trị thấp và ổn định.

- **Losses trên tập validation (val):**

**val/box\_loss, val/cls\_loss, val/obj\_loss:** Các giá trị loss trên tập validation cũng thấp và ổn định, không có dấu hiệu của overfitting nhiều.

### 3.3.3 Lần 3

- **Các thông số training**

- img: 640
- batch: 16
- epochs: 300
- weights: yolov5s.pt

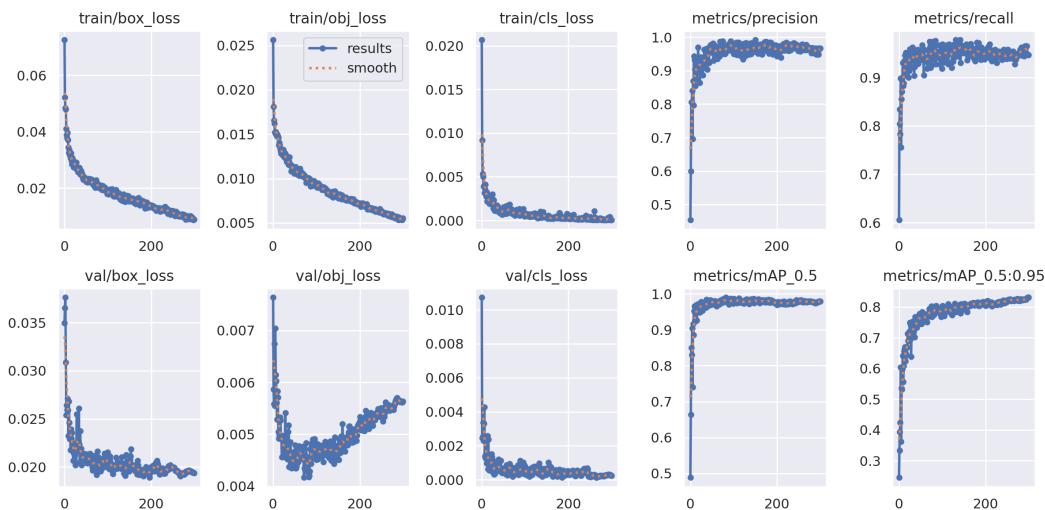
- **Kết quả training**

```

COMET INFO: Metrics [count] (min, max):
COMET INFO: loss [3030] : (0.10629470646381378, 3.001668930053711)
COMET INFO: metrics/mAP_0.5 [600] : (0.48851294901238074, 0.9903131588752223)
COMET INFO: metrics/mAP_0.5:0.95 [600] : (0.24542060767748067, 0.8311361560633859)
COMET INFO: metrics/precision [600] : (0.453650316572341, 0.9922947226241801)
COMET INFO: metrics/recall [600] : (0.6053671103477887, 0.9776877872524742)
COMET INFO: train/box_loss [600] : (0.008727232925593853, 0.0725418850779534)
COMET INFO: train/cls_loss [600] : (3.9887545426608995e-05, 0.020697979256510735)
COMET INFO: train/obj_loss [600] : (0.005329424981027842, 0.025628600269556046)
COMET INFO: val/box_loss [600] : (0.01888885162770748, 0.03764397278428078)
COMET INFO: val/cls_loss [600] : (0.00012785893341060728, 0.010728581808507442)
COMET INFO: val/obj_loss [600] : (0.0041670300997793674, 0.007634838577359915)
COMET INFO: x/lr0 [600] : (0.000166000000000000043, 0.07027272727272728)
COMET INFO: x/lr1 [600] : (0.000166000000000000043, 0.00990389696969697)
COMET INFO: x/lr2 [600] : (0.000166000000000000043, 0.00990389696969697)

```

Hình 3.6: Training lần 3



Hình 3.7: Đồ thị training lần 3

#### • Đánh giá kết quả training

##### – Loss:

**loss [3030]**: Dao động từ 0.1063 đến 3.0017, thấp hơn so với lần training trước và có vẻ ổn định.

**train/box\_loss, train/cls\_loss, train/obj\_loss**: Các giá trị loss đều thấp và giảm dần theo thời gian, cho thấy quá trình học diễn ra tốt.

##### – Metrics mAP (Mean Average Precision):

**metrics/mAP\_0.5**: Mean Average Precision với overlap threshold 0.5 nằm trong khoảng từ 0.4885 đến 0.9903, không có sự thay đổi lớn so với lần trước.

**metrics/mAP\_0.5:0.95**: Giảm từ 0.2454 đến 0.8311, nhưng vẫn ở mức cao.

##### – Metrics Precision và Recall:

**metrics/precision**: Nằm trong khoảng từ 0.4537 đến 0.9923.

**metrics/recall**: Nằm trong khoảng từ 0.6054 đến 0.9777.

Cả hai metrics này vẫn rất cao, thể hiện khả năng chính xác và độ phủ của mô hình.

##### – Learning Rates (lr):

**x/lr0, x/lr1, x/lr2**: Tất cả các learning rates vẫn có giá trị thấp và ổn định.

##### – Losses trên tập validation (val):

**val/box\_loss, val/cls\_loss, val/obj\_loss**: Các giá trị loss trên tập validation vẫn thấp và ổn định, không có dấu hiệu của overfitting nhiều.



## CHƯƠNG 4

### NHẬN XÉT MÔ HÌNH VÀ HƯỚNG PHÁT TRIỂN

#### 4.1 Những ưu và nhược điểm trong mô hình

##### 4.1.1 Ưu điểm

- Tốc độ xử lý nhanh, có thể xử lý với fps cao.
- Thu lại kết quả nhận diện vị trí và kích cỡ phương tiện chính xác.

##### 4.1.2 Nhược điểm

- Tốn quá nhiều tài nguyên tính toán nên việc thực hiện tính toán trên Google colab bị giới hạn, dẫn đến kết quả không mong muốn.
- Chưa thực hiện nhận biết tốt các phương tiện với mật độ lớn.
- Việc hiện thực vẫn chưa giải quyết được bài toán đếm phương tiện.
- Chưa phân biệt rõ các phương tiện khác loại

#### 4.2 Hướng phát triển trong tương lai

Dề xuất thêm giải pháp để xử lý vấn đề đếm phương tiện giao thông và tiến hành thực hiện nhận diện phương tiện giao thông trên thời gian thực, phục vụ việc thống kê lưu lượng xe dùng làm dữ liệu cho các nhà quy hoạch đường xá khắc phục tình trạng ùn tắc giao thông.



## TÀI LIỆU THAM KHẢO

- [1] Joseph Redmon, Ali Farhadi. University of Washington. "*YOLOv3: An Incremental Improvement*".
- [2] Joseph Redmon, Santosh Divvala, Ross Girshick, Ali Farhadi. University of Washington , Allen Institute for AI , Facebook AI Research. "*You Only Look Once: Unified, Real-Time Object Detection*".
- [3] Joseph Redmon, Ali Farhadi. University of Washington , Allen Institute for AI , XNOR.ai×. "*YOLO9000: Better, Faster, Stronger*".
- [4] Wikipedia. "*Convolutional neural network*".
- [5] Roboflow. "*Roboflow: Overview*".
- [6] Ultralytics. "*YOLOv3 Tutorial*".
- [7] Papers With Code. "*Average Pooling*".
- [8] Papers With Code. "*Mạng neural tích chập cheatsheet*".